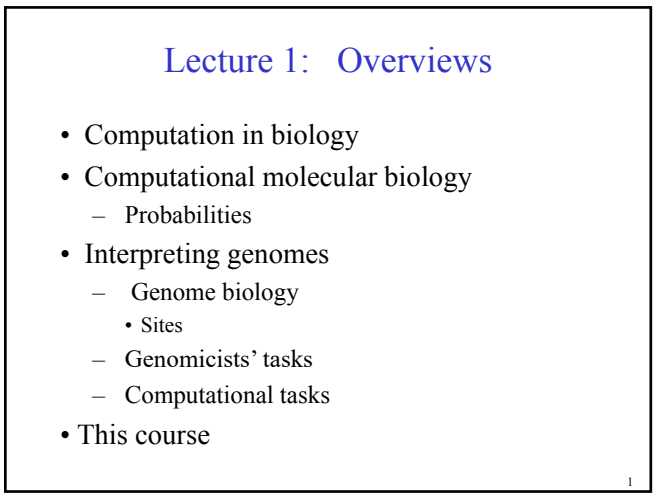


Hello, everyone, and welcome to Genome 540. I'm Phil Green, the instructor. In this opening lecture I'll give you some overviews that hopefully will help to put the course into context.

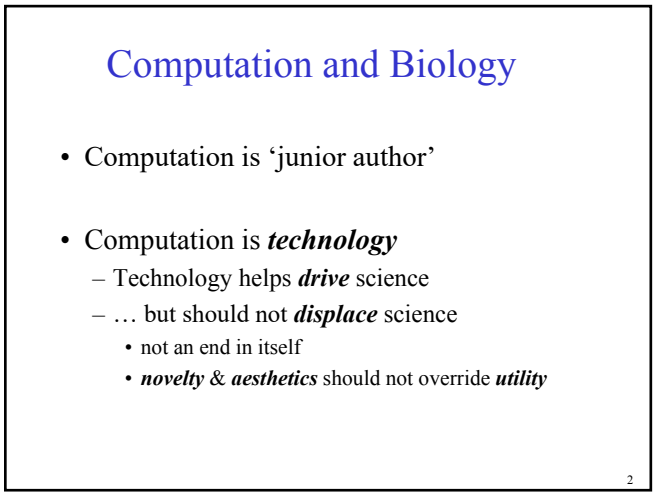
I'll start with some general comments about the role of computation in biology.

Then I'll give you my view more specifically of computational molecular biology, emphasizing in particular the fundamental importance of probabilities.

Then, because 540 is focused on computational methods for interpreting *genomes*, I'll present a high-level, somewhat non-standard view of genome biology which emphasizes ***sites*** — rather than, say, genes —as the fundamental units of functional information. This viewpoint turns out to be useful when we're developing probability models for the genome.

Next, a summary of genomicists' tasks and the roles that computation plays in those.

And finally, an overview of the course content and how it fits in with the computational tasks that you need for interpreting genomes, including some comments on what's not in the course and why.



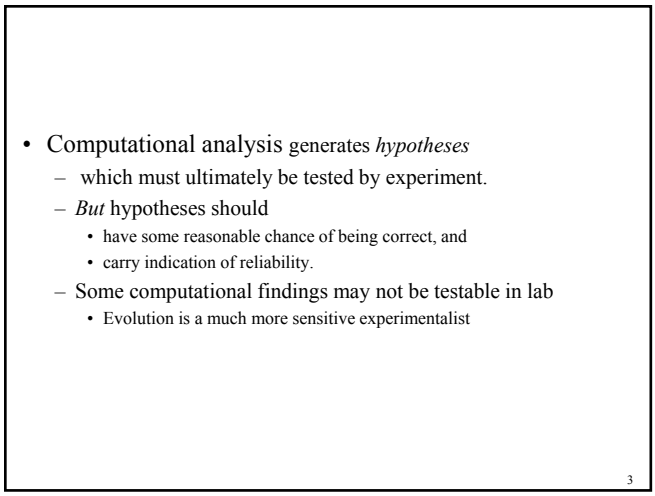
Let's start with three broad ways to think about the relationship between computation and biology.

One is that it's like the author list on a scientific paper. Biology is the senior author, which originates and motivates the work and provides the judgment to guide it to a conclusion. Computation is the junior author, which carries out much of the work, provides the energy and a lot of the ideas, but may lack the experience to know what's important and what's not.

If you want to do computational biology, you really should try to be like both authors. So in particular, if you come from a computational rather than a biological background, you should spend a fair amount of time trying to understand the biology: taking courses, reading textbooks and current literature, talking to biologists, and in general trying to develop an intuitive feel for the science — learning to think like a biologist. Even if you're not going to do research in biology it's still worth learning as much about it as you can, because it's an amazing, beautiful, and important intellectual achievement.

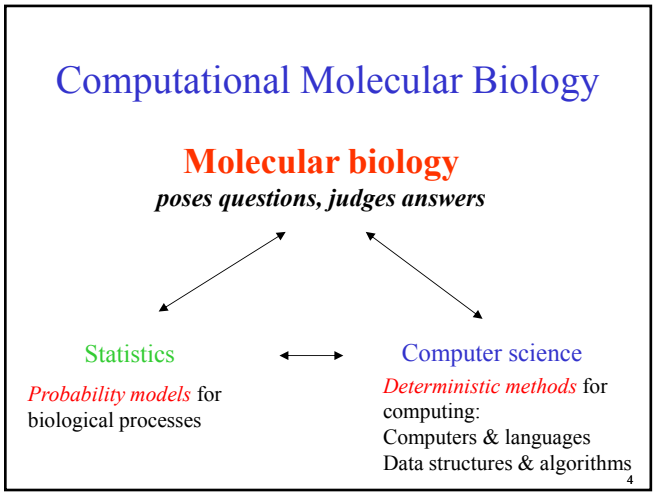
A second way of thinking about computation is as technology — like microscopes, or sequencing, or CRISPR, for example. It enables you to make scientific discoveries that would be difficult without it. And like these other technologies, computation actually alters the course of the science, by changing the kind of problem that scientists think about once they realize what it can do. In fact it's pretty clear you could not do biology as it's currently practiced without powerful computational methods. Analyzing, or even collecting massive data sets of the sort that are now central to molecular biology would not be possible.

At the same time though, technology should not *displace* the science: it's not an end in itself. People coming from computational backgrounds tend to prize novelty and aesthetics: a new, clever and elegant algorithm is something to strive for. But while those can be useful criteria, they should never override the utility. The purpose of the computational technology is to make biological discoveries, and that's frequently going to involve sacrificing both novelty — because an old method more often than not does the trick — and elegance, because the analyses needed may be 'kludges' stringing together a series of different steps. This may not be very satisfying to someone trained to appreciate the beauty of mathematics and computer science but it typically is just what you need to analyze biological systems (which are themselves kludges produced by a haphazard evolutionary process!)



A third way to think about computation is in terms of the role it plays in the scientific method. In general, computational analysis can't answer a biological question definitively, rather it generates hypotheses that need to be tested by experiments, according to the scientific method. You of course want these hypotheses to have some reasonable chance of being correct in order to persuade somebody — which could be yourself, or a collaborator, or some other biologist — to carry out the experiments. One of the reasons for using probability models (which we'll be discussing later in this lecture) for your computational analysis is that they allow you to make a strong case that a particular pattern is unlikely to be due to chance, and therefore is worth some experiments.

An interesting point here, though, is that sometimes experiments may not be practical, and computational evidence for a biological phenomenon might be the best you can do. This is because evolution can act on extremely subtle effects. For example, a mutation having a fitness effect size of 0.001 — which means the difference between leaving 999 rather than 1,000 descendants after some number of generations — would likely be very difficult to confirm in the lab, but is enormous from the perspective of evolution. In fact, population genetics theory tells us that a fitness effect size of the order of 1 / N, where N is the effective population size, is enough for evolution to go to work on. An experimental test of such an effect size (1 / N) would require you to work with the entire population of the species! Furthermore, most populations experience a variety of different environments, and it's unlikely you could reproduce all of them experimentally. So even much larger effect sizes may sometimes not be confirmable in the lab, if you can't reproduce the relevant environment. Evolution is a much more thorough experimentalist than humans can be, and for some of its experiments, computational analyses of the genome may provide our most convincing evidence.



Now let's focus a little more specifically on computational molecular biology (CMB) . I think of it as basically a convergence of three fields: *molecular biology* — that part of biology that tries to understand cells and organisms as systems of interacting molecules — and two computational fields: *statistics* and *computer science*.

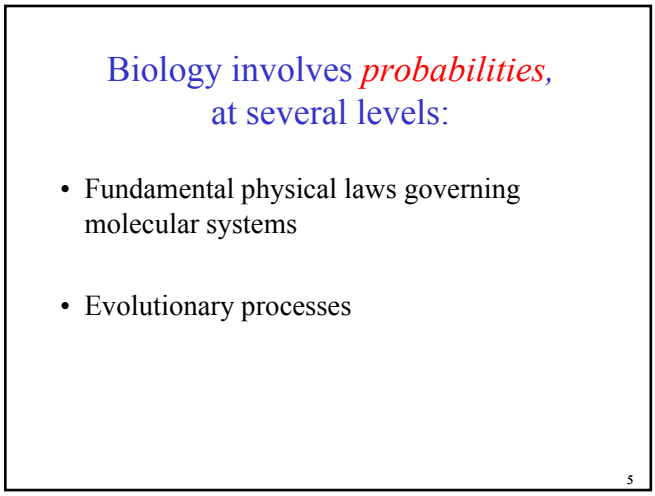
Of these, molecular biology is paramount: It poses the questions and judges the answers (like a 'senior author'). Whether or not your work as a computational biologist is worthwhile ultimately comes down to whether or not you're making a contribution to the biology.

The line between statistics and computer science has become increasingly blurred over the past few decades — computer science has become more statistical and statistics has become more computational— but there are still non-trivial differences in their perspective (and in most universities, they are still separate departments). The differences partly reflect a tension between what you might call 'fuzzy thinking' and deterministic thinking. The real world is 'fuzzy', in two ways: first, it's extremely complicated; and second, the underlying physical laws are in part probabilistic in nature (the same inputs can have different outputs). However, our brains construct simplified models of reality that are mostly deterministic, with causes and effects. Our scientific theories to some extent reflect this deterministic thinking but they also try to address the fuzziness.

In CMB, computer science comprises the deterministic aspects of computation — computers, programming languages, data structures, and algorithms— while statistics addresses the fuzziness, in particular contributing probability models for biological processes. By helping to simplify the biology and make it more manageable, such computational models play a role similar to that of experimental models such as model organisms and model systems.

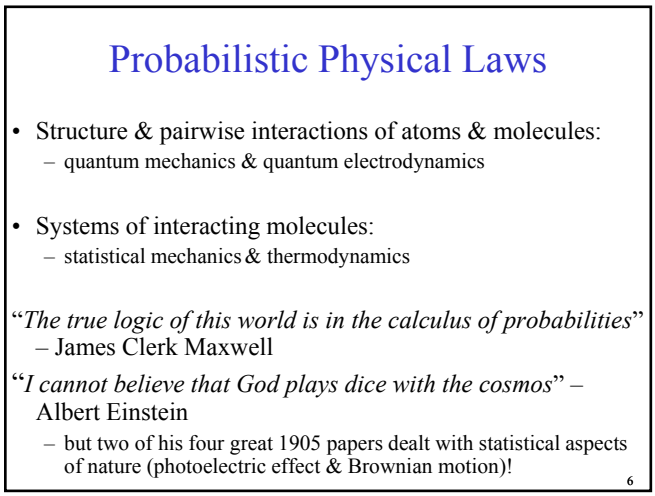
Because biological systems are much more complex than the systems that physicists and chemists tend to study, biologists have had to become more comfortable with 'fuzzy thinking', and consequently statistics in some ways has a closer relationship to the biology than computer science does. This close relationship goes back at least to the early 20th century, when both genetics and modern statistics were being developed, and a symbiotic relationship between them emerged: Geneticists needed statistics to interpret their data., and statisticians looked to genetics as a source of interesting problems. This was long before it was known that DNA was the genetic material, or electronic computers existed.

For computer scientists, fuzzy thinking is less congenial, and it can be a bit unsettling to learn that biologists don't even agree with each other on the definition of a gene — one of the most fundamental concepts in genome biology!



In the next lecture we'll talk more about algorithms, but here I'd like to say a little more about the role of probabilities in biology. Probabilities are important at two different levels:

First, at the fundamental level of physical laws for living organisms viewed as systems of interacting molecules, and second, at the higher level of evolutionary processes.



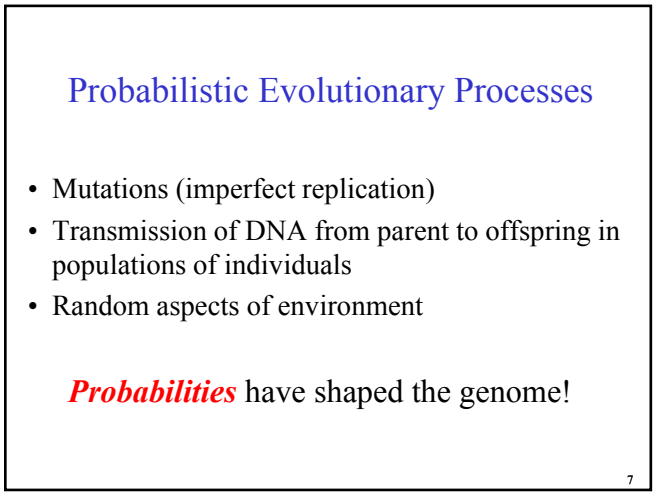
At the fundamental physics level you have, first, quantum mechanics and quantum electrodynamics, which determine the structure and pairwise interactions of individual atoms and molecules. In the prevailing 'Copenhagen interpretation' of quantum theory, the wave aspect of matter and radiation provides probabilistic information about particle locations, motions, and interactions

Systems of interacting molecules are complicated enough that there's no hope in practice of directly tracking the individual molecules (their coordinates, speeds, and so forth); rather, you have to understand the properties of the system by looking statistically at the ensemble of molecules. The relevant theory for this is statistical mechanics and thermodynamics. Again, fundamentally probabilistic.

These quotes from the two physicists generally considered the greatest since Newton bear on this issue of probabilities in physical laws. Maxwell is best known for putting the classical laws of electromagnetism in final form, the so-called Maxwell's equations. This was prior to the quantum era, and his laws are deterministic —they're about waves, but probabilities don't come into them. Nonetheless, he says "he true logic of this world is in the calculus of probabilities." Now, in fact, Maxwell was also one of the developers of statistical mechanics — among other things, he helped discover the so-called Maxwell-Boltzmann distribution of

velocities of molecules in gases — and this quote suggests that he regarded that work as perhaps more central to our understanding of how the world works.

On the other hand there's this contrasting quote from Einstein: "I cannot believe that God plays dice with the cosmos". Einstein did not like the idea of fundamental physical laws that were probabilistic in nature, and as a result, he never accepted quantum mechanics. Nonetheless if you look at the four great papers he published in 1905 (his 'miracle year') , although two of them (on special relativity and $E = mc^2$) had nothing to do with probabilities, the other two are both statistical in significant measure. One, on the photoelectric effect, helped instigate quantum mechanics; and the other explained Brownian motion of dust particles in a water drop under the microscope as the statistical effect of collisions with enormous numbers of water molecules (this work helped persuade many previously sceptical scientists that the atomic theory was in fact valid).



At a higher level, probabilities are important in evolutionary processes: in mutations as random changes to the DNA, transmission of DNA from parent to offspring in populations of individuals, inheritance of alleles (via chromosome segregation) and so forth. And then random aspects of a variable environment. So, since genomes are shaped by evolution, you can't really understand them without probabilities!

Genome biology overview

- Genomes undergo two fundamental processes (both involve copying!):
  - Replication
  - Transcription
- Genomic functional information is in the form of *sites:*
  - Short (~3 – ~15 base) sequence segments that bind to an *RNA* or *protein* molecule (the *reader*) to help mediate some function
- Sites may *act* (= be read) at the DNA or RNA transcript level

8

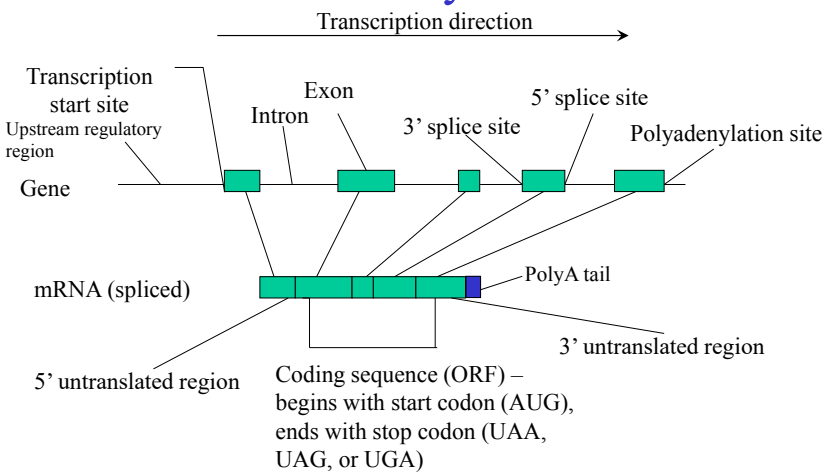
Now let's move on to interpreting genomes, starting with a high-level overview of genome biology. Genomes undergo two fundamental processes, both of which involve copying: *replication*, which is the copying of the entire genome into new DNA molecules, and *transcription*, which is the copying of parts of the genome into RNA transcripts.

The functional information in the genome is in the form of what I'll call ***sites***., which are short sequence segments (generally from about three to about 15 bases) that bind to an RNA or a protein molecule, which I'll call the ***reader***, to help mediate some function.

Sites can be grouped into two broad categories: Those that act (are read) at the DNA level, and those acting at the RNA level.

Now, those of you who've studied genome biology are probably wondering how I can talk about the functional information in the genome without mentioning g*enes.* One reason I'm not doing that here is the fact mentioned earlier, that the definition of a gene is not universally agreed upon; but a more important reason is that sites are really more fundamental. Genes, as well as other genomic features, are comprised of sites. And as we'll see, thinking in terms of sites is quite helpful in developing probability models of the genome. .

(Protein-coding) Gene Structure in Eukaryotes

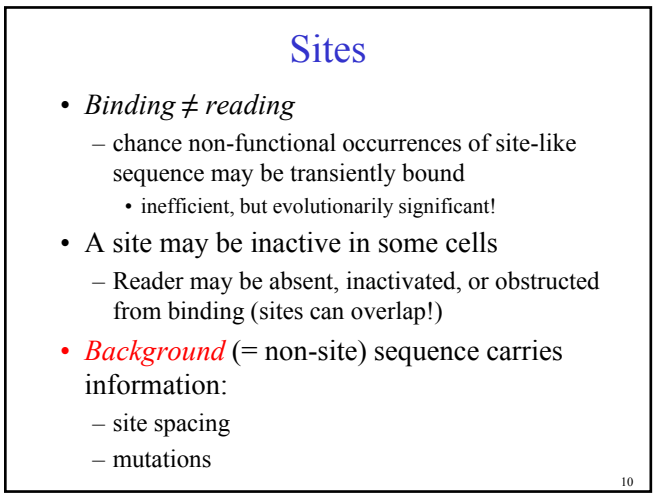


9

So we view a gene as a set of sites, and you can see a lot of them here. There are sites acting at the DNA level that control transcription. The unprocessed RNA transcript has sites acting at the RNA level for splicing out the introns and polyadenylation, and the processed transcript includes a 5' untranslated region with a translation start site and possibly some

translational regulatory signals; a coding sequence which is comprised of an array of codon sites; and a 3' untranslated region that may include, for example, microRNA binding sites and protein binding sites that play roles in targeting the transcript within the cell, controlling its degradation and so forth.

The ambiguity regarding the definition of a gene basically comes down to which sites do you choose to include in the gene and which do you not, but that becomes an uninteresting semantic issue once you focus on sites rather than genes as the fundamental units.



There are some subtleties in the definition of sites. One is that binding of an RNA or protein reader to some sequence is generally not sufficient in itself to make it a site; the binding event also has to help mediate some function within the cell. In general, that's going to involve the reader interacting with some other protein or RNA molecules to carry out some cellular process.

Site sequences are generally short enough that they occur frequently in random sequence. A transcription factor binding sequence for example might be just 6 or 7 bases, short enough that you can expect to find it by chance, every few thousand bases. Such chance occurrences may be recognized by a reader molecule and transiently bound without triggering any function — so aren't sites, by our definition. So there is presumably a fair amount of nonproductive binding to 'dummy sites'. But having short binding sequences also means that it's relatively easy to create new instances of potential sites via mutation. So from evolution's perspective, small size is a useful feature, rather than a bug. But it does increase the computational challenge of finding the functionally important ones.

A second point is that sites aren't necessarily active in every cell. The reader, or its interaction partners required to carry out some function, may not be available (not expressed, or inactivated in some way — for example, via phosphorylation, or by being bound to another protein that prevents binding to the DNA or RNA); or the reader may be present but prevented from binding because the DNA is methylated or already bound by some other protein (e.g. chromatin proteins, or readers at overlapping sites).

A third point is that although sites constitute the functionally important part of the genome, the non-site DNA (or ***background DNA***, as we'll sometimes call it) may still carry important information. In particular, the distance between nearby sites can influence interactions between reader molecules, which may be important for function. So the DNA that's between the sites may be important, not for its own sequence, but as a ***spacer*** for positioning the sites relative to each other. In addition, the background sequence is important for estimating mutation rates. That's useful even if you're only interested in the sites, because

one important way to detect sites is as regions that are relatively depleted of mutations due to purifying selection.

Sites: genomic distribution

- Sites typically *recur:*
  - multiple sites within a genome, with possibly varying sequences, may be recognized by the same reader
    - Sequence variation may be represented by a *motif* or (better!) a *sequence logo*
- Sites typically *cluster* (into '*features*'):
  - several sites, with the same or different readers, acting collectively to carry out a function
    - site *ordering, orientation* and *spacing* may be important
  - *gene* = cluster of sites involved in ***expressing*** a particular transcript

11

Sites are distributed non-randomly within the genome, something we'll need to take into account when developing probability models. First, sites recur, in the sense that a given reader will generally recognize multiple different sites. Once evolution has gone to the trouble of creating a particular reader, it tends to reuse it — for example, a given transcription factor is typically used in the expression of several different genes.

The different sequence instances of a site usually vary somewhat. The ***sequence logo*** (which we'll see an example of shortly) is one nice way of representing this variation in a manner that conveys frequency information. ***Motifs***, which are often used but less informative, indicate the possible nucleotides at each position but without frequencies.

Sites also typically tend to cluster (we'll call the clusters ***features***): several sites, with the same or different readers, may act collectively to carry out some function. Often there are positional constraints within the cluster. So for example, coding sequence is made up of multiple codon sites, and the positional constraints there are very strict since each codon immediately follows the previous one with no intervening bases and no overlap. Other constraints (for example, between splice sites) can be more lax.

A gene, as we saw before, is a cluster of sites involved in expressing a particular transcript. Expression of a protein coding transcript involves not only causing the transcription to occur, but also the processing of that transcript (e.g. splicing) and its translation into protein. So several steps are involved in getting to the end product, which is a protein molecule or molecules. There can also be additional steps in processing non-coding transcripts (for example, modification of nucleotides in tRNAs).

- Average site density (= the fraction of the genome that is functional) may be quite small!
  - < 10% of human genome
    - remaining > 90% mostly transposon relics, 'dead' genes & processed pseudogenes
  - strength of selection for 'genome efficiency' is expected to depend on
    - Population size
    - Reproductive life span
    - Genome size

12

How much of the genome do the sites represent? In bacterial genomes, that fraction seems to be quite high; typically, 70% or more of the sequence, might be protein coding, and when you add in RNA genes, transcription factor and other regulatory sites, and a replication origin, you're getting up close to 90% or more — not 100%, because there may be transposons and other parasitic DNA elements, and some DNA that's just playing a spacer role — but the vast majority of the genome does seem to be functional.

When you go to more complicated organisms, in particular the human genome, the situation is quite different. There are some 'intelligent design' proponents — including some scientists — who believe that either God, or evolution, has efficiently structured the human genome to be almost entirely functional. But the current prevailing view among most genomicists, based on comparing genomes to each other to estimate the fraction under purifying selection, is that only about 5% to 10% of the human genome is functional. However, a precise answer is hard to get, because of variability in mutation rates across the genome. and my own belief is that it's even less, around 2% — 60 million base pairs, or roughly 20 times the size of a typical bacterial genome. That's still a lot of DNA: After subtracting out the 35 million or so bases in protein-coding sequences and known functional non-coding RNAs, there's enough left to allow an average of about 1200 bases of regulatory sequence for each of the 20,000 genes — much more than has been found even for intensively studied genes.

Well, if the sites are less than 10%, what's the other 90% or more? At least 50% is identifiable as transposable elements, retroviruses, processed pseudogenes created by reverse transcription of RNA transcripts back into the genome, and 'dead' genes: sequences that look like they were once genes, but lack transcripts and have picked up enough mutations that they are clearly no longer functional. Much of the remaining 40% or so probably arose in the same way (from transposons etc) but over hundreds of millions of years has accumulated enough mutations to obscure the original source.

Why such a difference between human and bacterial genomes? There are several reasons why selection for efficient genome organization ought to be much stronger in bacteria than in humans. One factor is relative population sizes. As I mentioned earlier in the lecture, according to population genetics theory evolution acts on fitness differences as small as 1/N where N is the effective population size of the organism, so more sensitively for organisms with large populations. Humans, for most of their history, seem to have had a fairly small effective population size numbering in the tens of thousands, far less than bacteria.

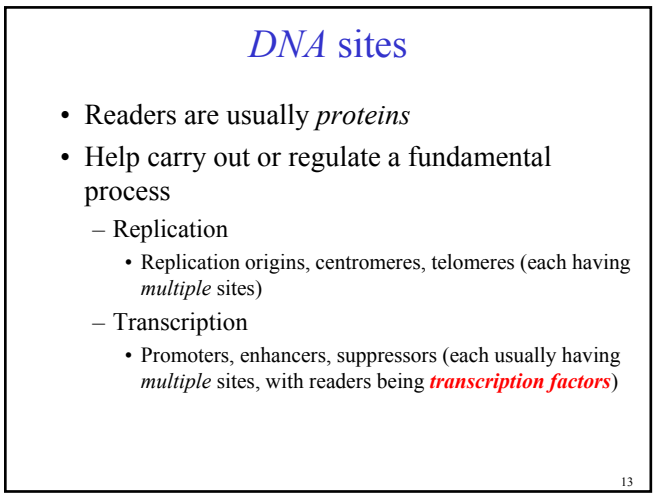
Another factor is reproductive lifespan. Many bacteria grow fast enough in nutrient-rich conditions that replication of the DNA is rate-limiting, and one expects there to be intense

competitive pressure to keep replication time, and therefore genome size, as small as possible.

Finally, as the genome gets larger, each added transposable element (for example) represents a diminishing percentage of the total size, and so one expects selection against it to correspondingly decrease. The human genome, at 3 billion bases, is 1,000-fold larger than a 3 megabase bacterial genome, and an added transposon that doesn't interfere with site activities has proportionately smaller impact .

Consistent with the above, other eukaryotes such as yeast, *Drosophila*, *C. elegans,* fish, tend to be intermediate between bacteria and humans with respect to population size, life span, and genome size, and they're generally also (as predicted by the above considerations) intermediate in the estimated functional proportion of the genome.

Also consistent with this idea that selection for efficiency has been weak in humans is the finding that in many human cells a surprisingly high fraction of newly synthesized protein molecules misfold and then are immediately degraded. This is a major waste of cellular energy, probably, in fact, much worse than the energy lost in replicating an unnecessarily large genome.



Recall that sites may act either at the ***DNA*** level or at the ***RNA*** level. The DNA level sites usually have protein readers, and they help carry out or regulate one of the two fundamental processes, replication or transcription.

Replication-associated *features* (site clusters) include replication origins, centromeres, and telomeres. I include telomeres here because one of their major roles is to ensure the faithful replication of the ends of linear chromosomes; and centromeres, because they are involved, not in DNA replication *per se*, but in ensuring the faithful distribution of the products of DNA replication to daughter cells.

Transcription involves several types of feature: promoters, enhancers, and suppressors. The readers in this case are called ***transcription factors***.

From http://www-lmmb.ncifcrf.gov/~toms/sequencelogo.html

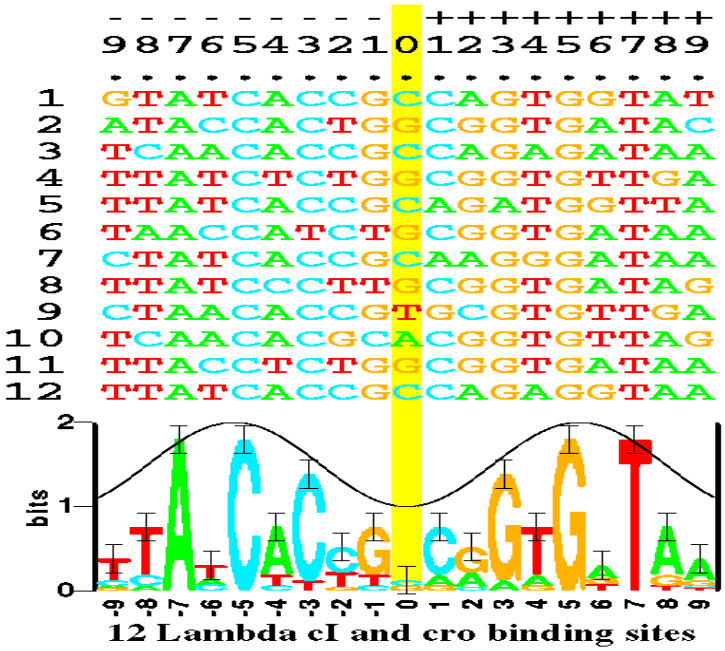


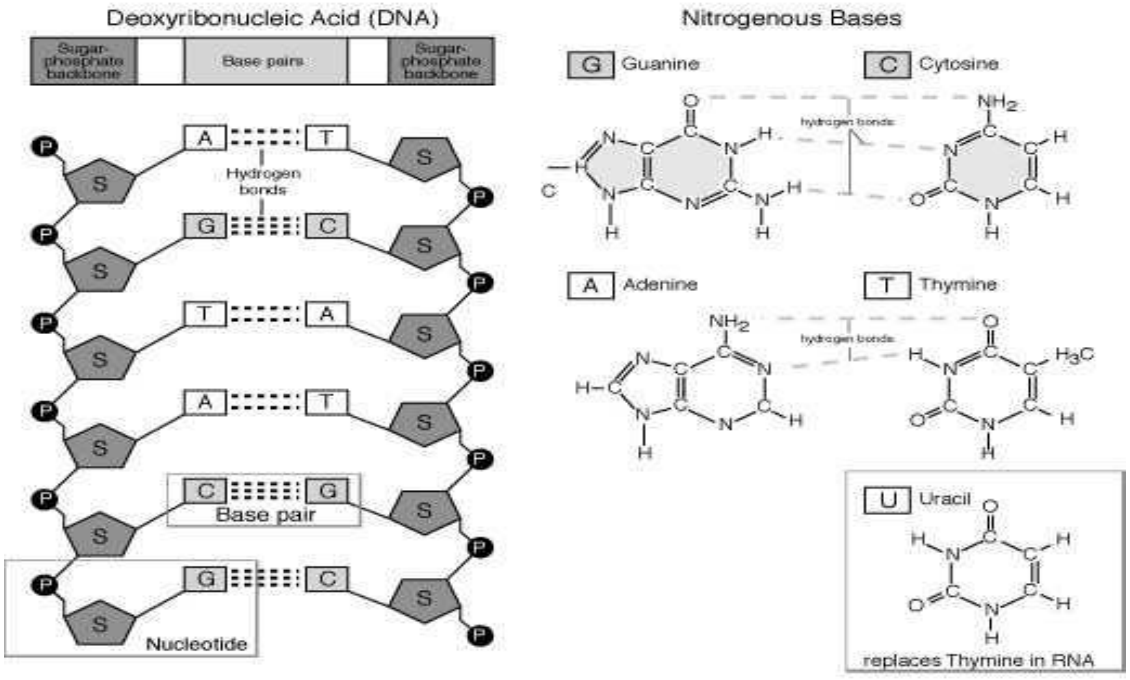
Fig. 1. Some aligned sequences and their sequence logo. At the top of the figure are listed the 12 DNA sequences from the P_L and P_R control regions in bacteriophage lambda. These are bound by both the cI and cro proteins [16]. Each even numbered sequence is the complement of the preceding odd numbered sequence. The sequence logo, described in detail in the text, is at the bottom of the figure. The cosine wave is positioned to indicate that a minor groove faces the center of each symmetrical protein. Data which support this assignment are given in reference [17].

14

Let's look at an example of a transcription factor binding site. This figure is taken from the website of Tom Schneider, who invented sequence logos of the sort depicted here at the bottom. The sequences are from the genome of a bacterial virus (or phage), *lambda*, which infects *E. coli* . Each sequence consists of a cluster with two binding sites, each of length 9 bases, for the transcription factor Cro (a different transcription factor, cI, also recognizes these sites). There's a 1-base spacer (always the same size!) between the two sites, so the total length of each cluster sequence is 2 x 9 + 1 = 19 bases.

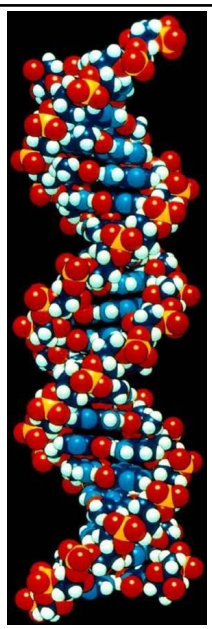
There are 12 different sequences here, but they correspond to only 6 different clusters in the lambda genome, because both DNA strands are given for each cluster. So sequence 2 here is the reverse complement of sequence 1, 4 is the reverse complement of 3 and so on.

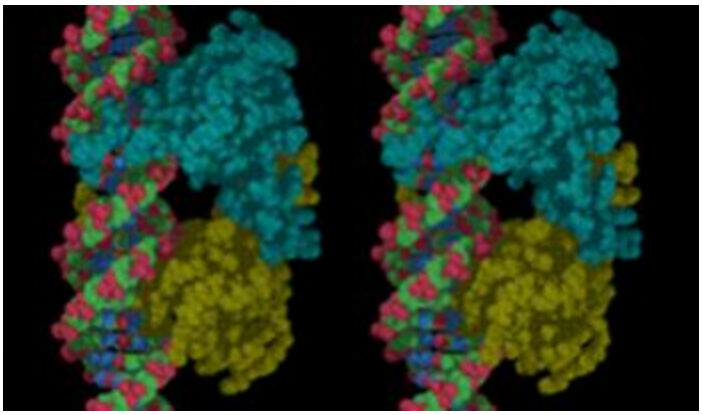
National Human Genome Research Institute (NHGRI) http://www.nhgri.nih.gov/DIR/VIP/ by artist Darryl Leja



15

To fully understand what's going on here, we need to picture the protein binding to the DNA in three dimensions. First, recall the two-stranded molecular structure of DNA. This slide shows it schematically on the left, indicating the phosphate-sugar backbone on the outside and the A:T and C:G base pairs on the inside. The two strands run in opposite directions: the strand on the left has its 5' end on the top and its 3' end on the bottom, whereas the strand on the right has them reversed (the sugars are upside down).





from http://gibk26.bse.kyutech.ac.jp

from http://www.dna-dna.net/

16

Now, on the left above is a space-filling model of the atoms in a DNA molecule. Note first of all that it is a double helix, with the two strands winding around each other and base-pairing with each other. These two ridges are the sugar-phosphate backbones of the two strands, and the base-pairs you can see to some extent in the grooves. There are two continuous grooves: the so-called major groove here, which goes around the back and re-emerges here, and the minor groove here, going back behind and coming up here. Every base shows up partly in the major groove and partly in the minor groove. Although you can't really make out the base pairs, each full turn of the helix — so for example from a point here to a point here, or a point here to a point here — corresponds to about 10.5 base pairs.

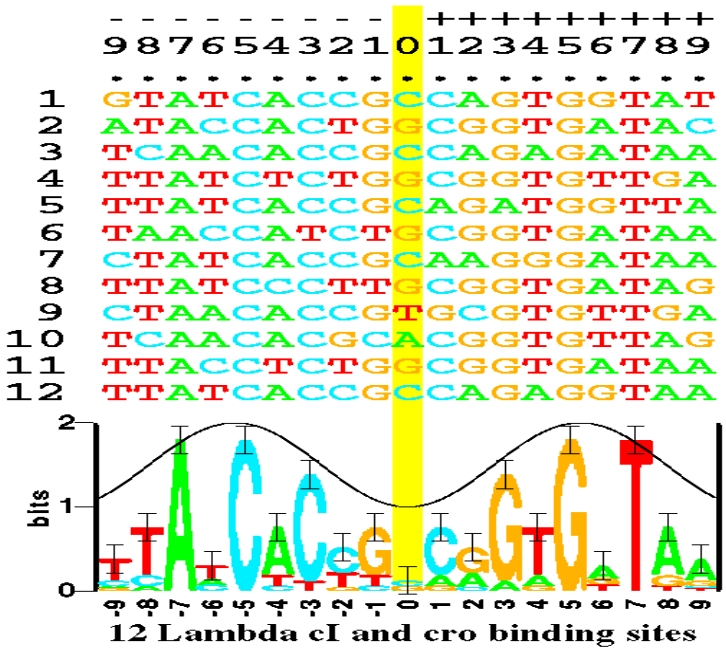
Note also that the helix is right-handed, meaning that if you point the index finger of your right hand along a ridge of the helix, your thumb points in the vertical direction (up or down) that the ridge is going. If you try that with your left hand instead, the thumb points in the wrong direction.

Most transcription factors bind primarily in the major groove, because there are more opportunities to make contacts with atoms in the nucleotide bases there, although there are some that bind within the minor groove, or within both grooves.

Above on the right is a stereo image: if you cross your eyes to make the two images converge you can see this in 3D. What's depicted is two identical copies (one in blue and one in yellowish-green) of one of these proteins (cI or cro, I'm not sure which) binding to

the double helix at one of the sequences depicted on the earlier slide. Both are making contact with the major groove but also with each other, and in fact that interaction with each other helps to increase the overall stability of both molecules binding to the DNA sites, so it's important. For this particular protein, the contact between the two copies requires one to be flipped around with respect to the other. That means that the DNA sites that they bind to are also flipped around (in reverse orientation).

From http://www-lmmb.ncifcrf.gov/~toms/sequencelogo.html



12 Lambda cI and cro binding sites

Fig. 1. Some aligned sequences and their sequence logo. At the top of the figure are listed the 12 DNA sequences from the P_L and P_R control regions in bacteriophage lambda. These are bound by both the cI and cro proteins [16]. Each even numbered sequence is the complement of the preceding odd numbered sequence. The sequence logo, described in detail in the text, is at the bottom of the figure. The cosine wave is positioned to indicate that a minor groove faces the center of each symmetrical protein. Data which support this assignment are given in reference [17].

14

Now let's return to the multiple site sequences. Recall that there are 6 different clusters, and the two sites in each are in opposite orientations so that the two protein molecules can contact each other. To compare all twelve sites, you need to include the reverse complement sequences so that the right hand sites are put in the same orientation as the left hand ones. Note that for convenience each site is represented by a single-stranded sequence, but the protein itself contacts both strands simultaneously.

So on the left side here you have the 12 9-base sites all in the same orientation; on the right you have the same 12 sites but now all in the reverse orientation. So the sequences on the right (and the overall patterns) are the reverse complements of what's on the left.

Note that the site sequence is not invariant. At the bottom is a ***sequence logo***, which reflects the frequencies with which different nucleotides are used at each position. In all of these sites you have a C here, and an A here; these presumably correspond to the most important contact points with the protein; in the reverse complements you have a G here and a T here (the complementary nucleotides, in the reverse order). Some of the other

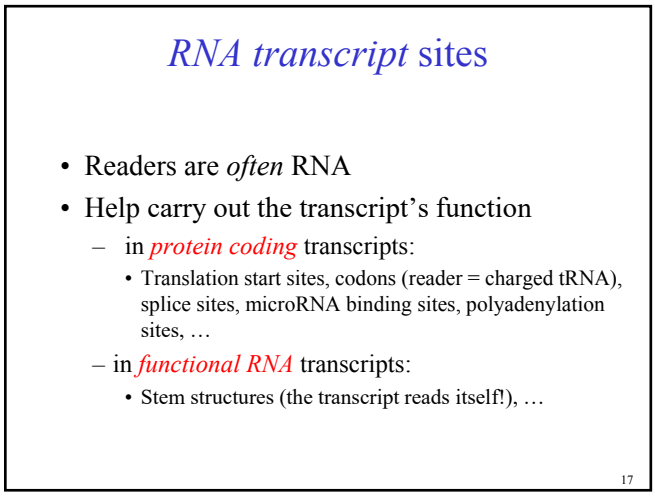
positions appear to show somewhat weaker conservation and others appear free to vary (including, as you'd expect, the spacer nucleotide between the two sites).

The logo is actually comparing two probability models, one reflecting frequencies with which nucleotides are used in instances of the site, and the other the so-called background probability model which gives the average frequencies of nucleotides at non-sites (or the genome as a whole). The total letter height at each position indicates how much better the site model fits than the background. In a sense this corresponds to the information the protein needs in order to pick out a site from the rest of the genome.

The biology underlying site sequence variability is interesting and not always understood. Some variability is presumably at positions where contact with the protein is weak or non-existent, and so has little impact on the strength of binding. At positions where contact does occur, some variability might be important in regulating the strength of binding (some sites might need to be bound more tightly than others).

The curve that's depicted here has a spacing of 10.5 bases between peaks, and is intended to reflect the fact that the proteins are binding on one side of the helix so you might expect the contact points in the two site copies, and the most highly conserved bases, to show that spacing. That seems to be approximately true here.

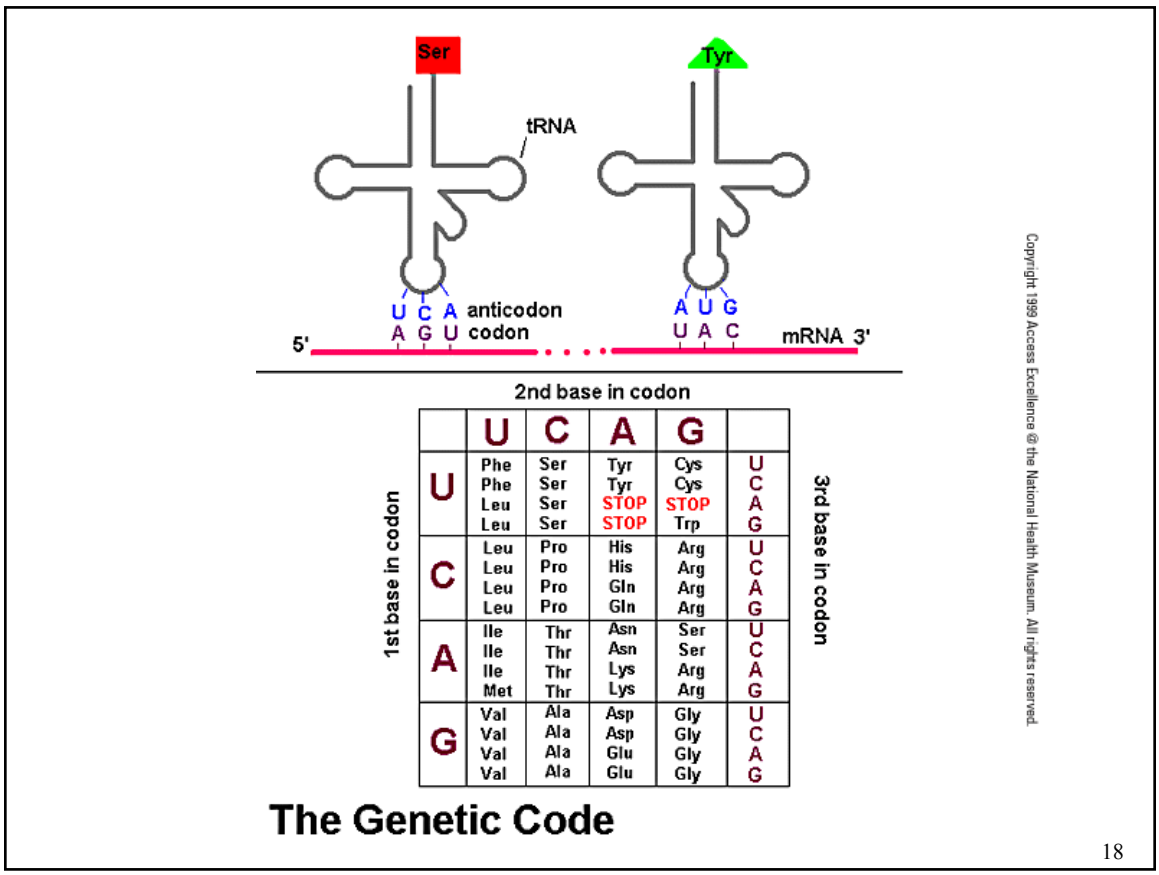
Another thing to note is that the total number of conserved positions in one site is small; only about 3-4 bases — so it makes sense that you need a cluster of two of them to get some specificity and stable binding.



The second broad class of sites are those acting at the RNA level, with the reader recognizing the site within a transcript (not the genomic DNA) and thereby helping to carry out the transcript's function. (Of course the site's sequence is also present within the genomic DNA, since the transcript is a copy of it). Often, but not always, the reader is itself an RNA transcript.

As we saw earlier, protein coding transcripts contain a variety of sites.

There are also RNA transcripts that don't encode proteins but carry out some other function in the cell -- for example tRNAs, ribosomal RNAs, spliceosomal RNAs, microRNAs, and a variety of so-called lncRNAs (long noncoding RNAs). These all contain at least one type of site that might seem a little strange, but fits the definition I'm using: namely 'stems' that basepair one short sequence within the transcript to a complementary sequence within the same transcript, and which thereby help to give the transcript a structure that is important to its function. So these are sites for which the transcript is reading itself!



	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

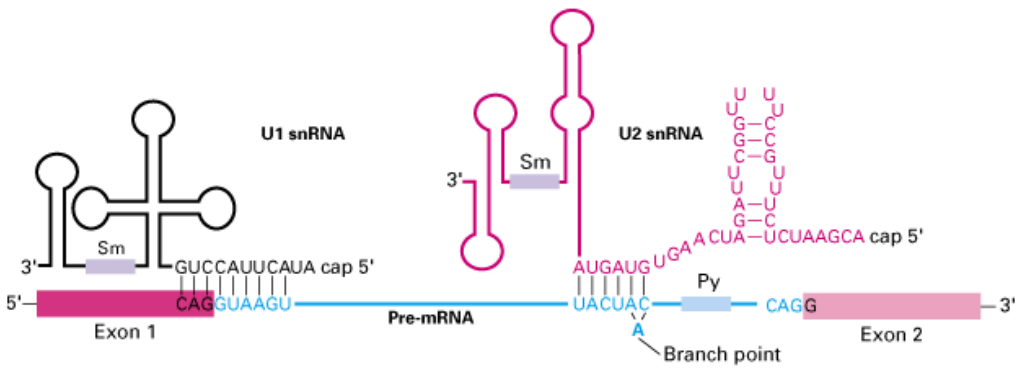
Here's an example illustrating some RNA sites involved in protein translation.

Within a protein-coding transcript the codons are 3-base sites whose readers are tRNAs. In this case the codon AGU, which you look up in the table as the first base A, second base G, third base U, so it's a serine codon. The reader is a tRNA ('charged' with the amino acid serine) that has the complementary 'anticodon' sequence ACU (in 5' to 3' order).

Further, the tRNA, as a functional RNA in its own right, has a stem structure with sites where the RNA base pairs internally to itself; and also additional sites, one of which is recognized by the tRNA synthetase protein which covalently attaches a serine molecule to the tRNA, and others which are recognized by proteins that modify some of tRNA's nucleotides to stabilize it.

Serine has five additional codons in this table that are read by other tRNAs. Some tRNAs can read more than one codon — in fact I think this tyrosine tRNA can read both of the tyrosine codons, via wobble pairing — so there is sequence variability for some of these codon sites similar to what we saw in the transcription factor case.

Of course the translation process involves this whole complex (charged tRNA + mRNA codon) interacting with the ribosome and there are other sites in the tRNA and the coding transcript that are recognized by various proteins within the ribosome.



from http://departments.oxy.edu/biology/Stillman/bi221/111300/processing_of_hnrnas.htm

(Jonathon Stillman, Grace Fisher-Adams)

19

Here's another example: sites involved in the splicing process. First, the so-called 5' splice site (meaning that it is at the 5' end of the intron). The reader is the U1 small nuclear RNA that recognizes it by base pairing. This picture shows perfect pairing, but in fact typically the pairing is not perfect, that is you don't have an absolutely required base at all these positions within the transcript. So the U1 RNA sequence here can base pair with multiple possible sequences here, and you have sequence variability as in the previous examples. The U1 RNA has its own sites, including the stem structures that stabilize it, sites that interact with protein components of the spliceosome, and so forth.

A little upstream of the 3' end of the intron there's a ***branch site*** which is recognized by the U2 snRNA. And then at the 3'end are other sites that are recognized by *protein* components of the spliceosome. There are some constraints on site spacing but they are quite weak, as indicated by the fact that intron sizes vary enormously. If the intron is too small, the splicing machinery either can't recognize the intron at all or it can't process it correctly, so there is a lower bound on intron size (about 70 to 80 bases in human genes, with rare exceptions). But there's no corresponding upper limit and there are introns that are hundreds of kilobases long.

Genomicists' tasks

- Find the ***genome sequence***
- Find the ***transcripts***
- Find the ***sites*** ...
- … and their ***functions*** …

20

Given that view of the biology, in order to interpret genomes, genomicists first need to get the genome sequence, and identify the transcripts that are made from the genome. Then they have to find the sites, being mindful that sites can act either at the DNA or at the transcript level. On the following slides I'll say a bit more about each of these.

Finally they have to illuminate the molecular functions of the sites. This is really the most open-ended and difficult part, requiring a variety of methods, and I won't try to cover it. One thing that's very helpful though is the fact that sites recur, not only within one organism's genome but also between the genomes of different organisms, and so once you've figured out the function for a particular site you largely understand the function for many other occurrences of that site as well.

Finding the genome sequence

- Get *reads* (short, overlapping, error-prone pieces of the sequence)
- *Assemble* : identify read overlaps, infer underlying sequence
- Main challenge:
  - (Near-)duplicate sequences

21

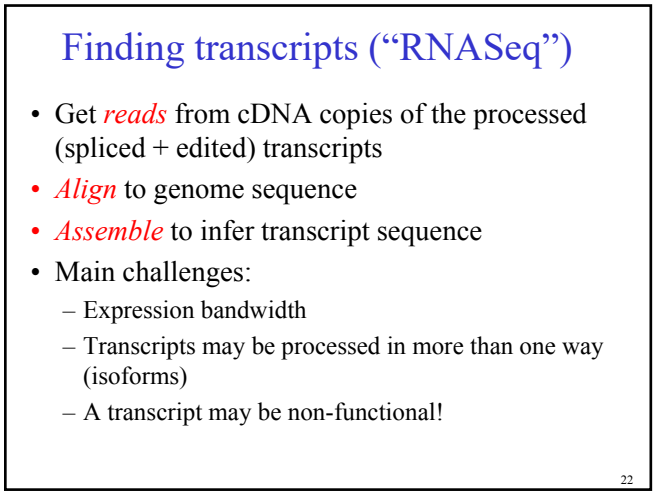
The main approach to finding the genome sequence requires getting ***reads***, which are the sequences (often with basecalling errors) of short pieces of the genome, and then assembling those to infer the underlying genome sequence. The assembly process involves, in essence, finding sequence matches between portions of the reads, figuring out from these how the reads overlap in the genome, and then piecing the overlapping reads together while identifying and eliminating basecalling errors in order to reconstruct the underlying genome sequence.

The main challenge in assembly is duplicate or nearly duplicate sequences within the genome, which arise in evolution in several ways. One type is self-copying parasitic DNA elements (such as transposons), which typically are a few hundred to a few thousand bases long. Another type is segmental duplications arising from errors in DNA replication, which can be up to several megabases.

Consequently, when two reads have portions that are highly similar, one has to consider the possibility that they do not actually overlap within the genome but rather come from different copies of a duplicated segment. The different segments often have acquired sequence differences via mutation that in principle should help to identify spurious

overlaps. But read basecalling errors complicate this, and evolutionarily recent duplicates can be essentially identical.

A variety of assembly strategies have been developed that try to cope with the duplicate segment issue. Computationally, it helps to have probability models for both basecalling errors and the mutation process. But the 'killer technology' finally allowing assembly of essentially complete human genomes has been the advent of very long reads (longer than most duplicated segments). Even with those however comparing reads to each other remains an important requirement.



Now, finding the transcripts, sometimes called ***RNASeq***. Since it's easiest to sequence DNA, RNASeq involves first of all making cDNA copies of the processed mRNA transcripts using reverse transcriptase, and then getting sequence reads from the cDNA. Typically these reads are then aligned to the genome, which allows targeted assembly to be done for reads mapping to the same genomic region in order to reconstruct full transcript sequences from the region.

There are a number of issues here that collectively make finding all the transcript sequences a more challenging problem than sequencing the genome.

One is that many transcripts can be spliced in more than one way ('alternative splicing'), resulting in multiple ***isoforms*** that may encode somewhat different proteins. Different isoforms share parts of their sequences with each other, which presents an assembly problem similar to that presented by duplicate segments in genome assembly.

Another issue is ***expression bandwidth***: genes may be expressed at very different levels, with some transcripts several orders of magnitude more frequent than others. This greatly increases the amount of sequencing that must be done in order to be sure of getting the rarest transcripts. That issue doesn't really arise in genome sequencing since all portions of the genome are equally represented in the starting DNA from which libraries are made (although library construction can sometimes introduce biases!)

Another issue is that expression level, and to some extent splicing, depends on the cell type. So you have to make cDNA libraries from many different cell types to maximize the chance you're getting all isoforms of all genes.

Yet another problem is that at least some transcripts are non-functional. Some transcripts from protein-coding genes result from splicing errors (which are common enough that there is a cellular process, nonsense-mediated decay, for detecting and degrading them). Most lncRNAs, which by definition lack protein coding potential, also currently lack any other known function within the cell. Novel functions have been discovered for a few of

them, and may be for others. But given that selection for efficiency appears to be relatively weak in the human genome, it's also quite plausible (even likely!) that most lncRNAs are simply transcriptional 'noise'. Alternatively many of them may be byproducts of transcriptional events in which the act of transcription is important because it remodels the chromatin (which may be important for expression of nearby genes), but the transcript itself is non-functional. In such a situation the functional sites of interest would be the transcription-inducing sites acting at the DNA level, and not RNA-level sites in the transcript.

Finding sites

- Direct detection of binding events (e.g. ChIPSeq)
  - ***but*** binding may be non-functional!
- Computational search for clusters of recurring motifs
  - ***but*** motifs occur frequently by chance, in any large genome!

23

A harder problem is to find the sites. One method that a lot of work has gone into, for example in the ENCODE project, is the direct detection of binding events. One approach for this is to use antibodies (or some kind of tagging) to a particular transcription factor to isolate that factor bound to DNA and then sequence the DNA. A similar strategy could be used for other DNA binding proteins, and presumably also for RNA-binding proteins. This approach seems to be limited to readers that are proteins, and it requires some knowledge of what they are; but a more serious objection is that, as we discussed earlier, you can have binding without it being functional so the sequences you get may include non-sites.

A somewhat complementary computational approach is to look for clusters of recurring motifs, not only known ones from the binding studies but also novel ones which have similar lengths, distributions of conserved positions, nucleotide composition, and clustering patterns to the known ones. Because site motifs are short they occur often by chance, so small clusters (of 1 or 2 sites) may not be reliably detected.

Compare genomes of …

- a lab organism & a singly mutated variant with an altered phenotype
  - the mutation must then alter (or create!) a site
    - or alter site spacing
  - and the phenotypic change illuminates its function
  - but remember that cells with identical genomes can sometimes have different phenotypes!
    - Tissues in multicellular organisms
- members of a natural population
  - Usually *multiple* genomic and phenotypic differences
  - find correlations (of *recurring* differences) to identify sites that affect a particular phenotype.

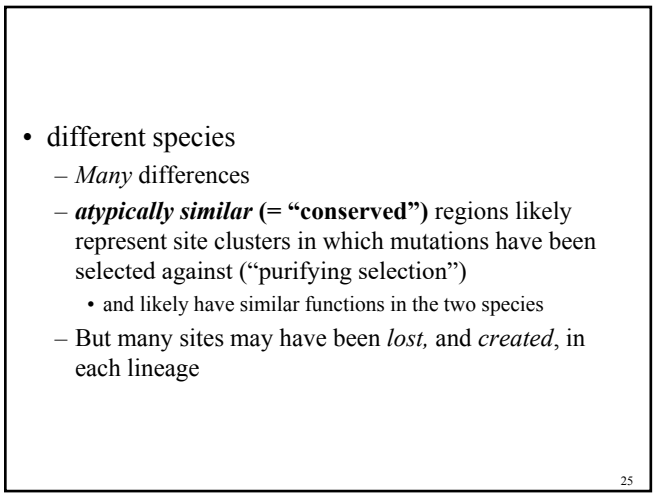
24

So both of those methods are error-prone to some degree. A different and generally more definitive strategy (although with its own limitations!) is to compare genomes that differ from each other and try to relate the sequence differences to phenotypic differences (in physiology, or other organismic or cellular characteristics). This can point you both to the sites and, sometimes, to a functional role for specific sites.

The key point is that a difference in phenotype usually means that there is a sequence difference affecting sites. For phenotypes at the cellular level you have to be a little careful because of the fact mentioned earlier, that for a variety of reasons sites may vary in activity across cells and so cells with identical genomes can nonetheless have different phenotypes. But assuming you can control or check for that, phenotype differences usually imply sequence differences that either alter a site's activity or create a new site. Insertion or deletion mutations in background sequence between two sites could also have a phenotype by altering site spacing, but that's not an issue with point mutations.

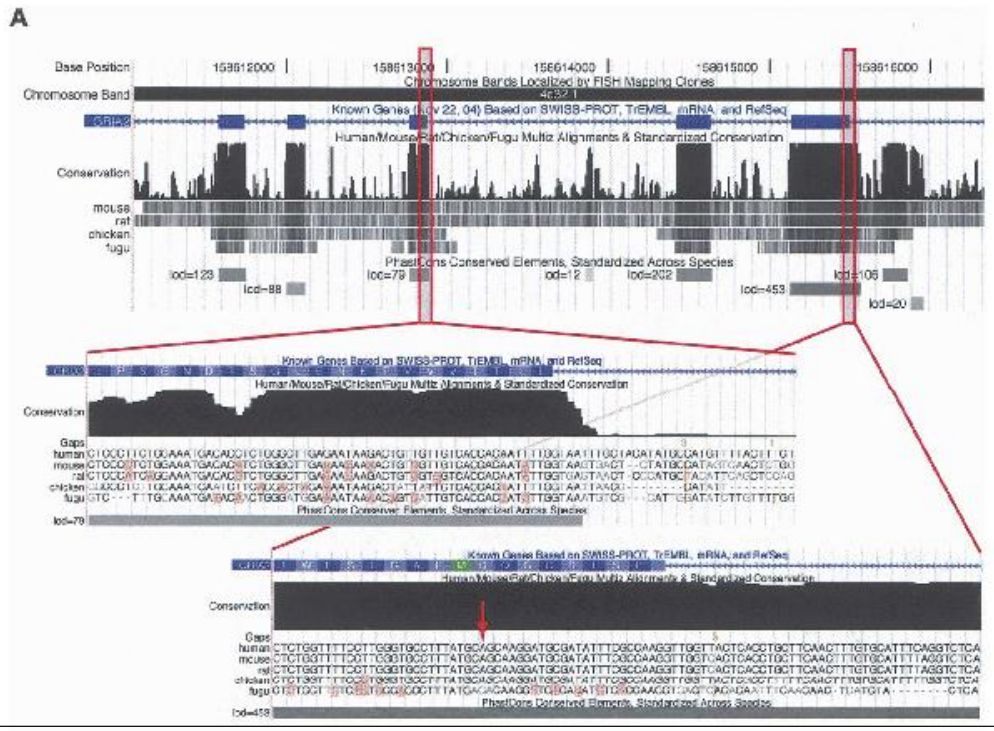
In experimentally manipulable organisms (or cells) you can in principle find sites using CRISPR (for example) to systematically make mutations and assess phenotypes. Of course this is quite challenging on a genome-wide scale, and more seriously it may not actually find all sites because (as previously discussed) our own ability to assess phenotypes in the lab is much less sensitive than evolution's.

Another approach is to leverage naturally occurring mutations by comparing different members of a population. The problem here is that generally any two individuals have multiple sequence differences and multiple phenotypic differences, so associating phenotype to genotype is challenging. What you have to look for is correlations of recurring differences — shared phenotypes between individuals having shared genotypic changes. GWAS studies do this in a targeted way. But GWAS typically does not, at least by itself, pinpoint the affected site because genomic variants in linkage disequilibrium with each other often have similar correlations with the phenotype.



Finally, you can compare different species. In this case you will have many sequence differences and many phenotype differences, the numbers depending on how closely related the species are. We again expect phenotype differences to largely reflect differences in site content, i.e. sites present in one organism but absent from the other. Conversely, shared sites should typically correspond to shared aspects of phenotype.

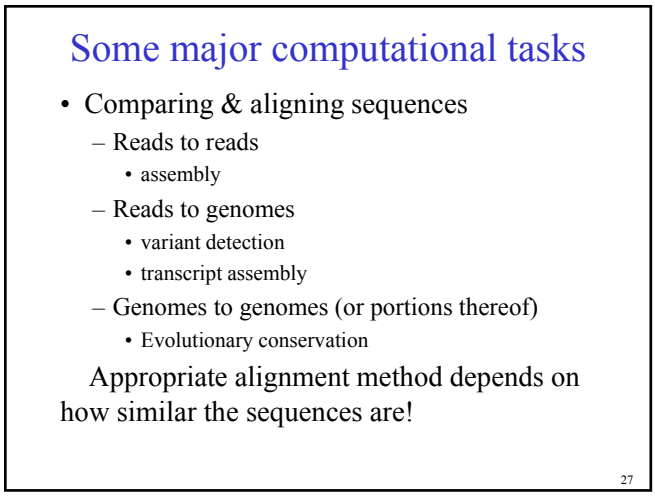
Since background (non-site) sequence is not under purifying selection, it accumulates mutations more rapidly than site sequences do. For very distant organisms, the density of accumulated mutations may make alignment of background impossible; for closer organisms, it may be possible, but shared sites should still be detectable as having higher similarity (greater ***conservation***, due to purifying selection) between the species than the background. So in both cases the alignment can give us information about shared sites, but it says essentially nothing about lineage-specific sites underlying the differences between the species.



from Siepel A. *et al.* (2005). Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome Res.* 15:1034-50.

26

Illustrating that, this figure from Adam Siepel's paper on his program ***PhastCons*** depicts alignments of a particular region in the human genome to four other vertebrates: mouse, rat, chicken and fish. Down here we see the sequence, up here a schematic indicating aligned segments. You can see that the alignments to chicken and especially to fish are pretty much confined to exons in the genes in this region, with a little bit of surrounding sequence that may include some regulatory sites. The alignments to our fellow mammals mouse and rat are much more extensive, which could be partly due to shared mammalian-specific sites but is mainly due to the background sequence still being alignable. Within the aligned portions there are some regions, almost certainly clusters of shared sites under purifying selection, that stand out as having a much higher degree of conservation, indicated by these bars here.

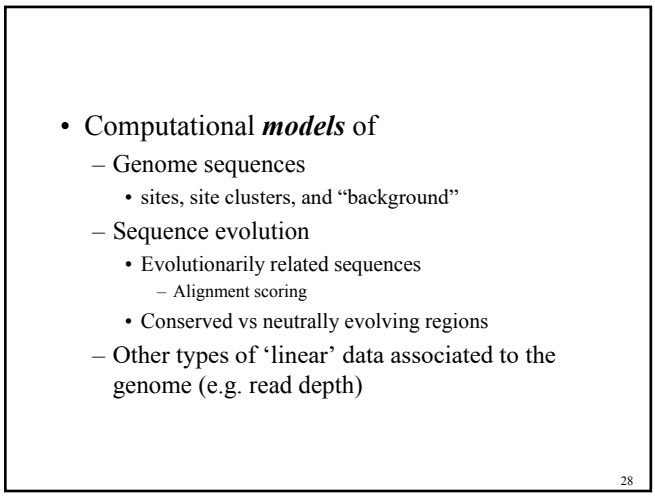


Hopefully, you'll have noticed in the preceding survey of genomicists' tasks that there are some recurring computational themes.

One is ***comparing*** (and ***aligning***) sequences: Sequence assembly involves comparing reads to reads; variant detection and transcript assembly involve comparing reads to genomes; and finding evidence of shared sites between species involves comparing genomes to genomes.

The appropriate alignment method depends on how similar the sequences are. In these first two cases the sequences are highly similar to each other, with isolated single-base differences from base-calling errors and mutational variants, and (in the case of transcripts aligned to the genome) isolated large gaps corresponding to introns. In such cases, methods of the sort we'll discuss in the next lecture allow you to quickly find large perfectly matching segments which can then be easily extended to alignments.

However the last case, involving distantly related genomes, or genes, requires more sensitive methods which we'll discuss later in the course.



A second recurring computational requirement is ***models*** — simplified representations of the genome sequence, sequence evolution, and alignments that can guide computational analyses. To computationally find sites within genome sequences, we'll want to model sites and site clusters, as well as the non-site background. To find shared sites within genome alignments, we'll want to computationally model mutation and purifying selection. In addition, to find the alignments themselves we'll need a model to tell us how to do the scoring of an alignment.

It will turn out that the models we develop for the above purposes are helpful in analyzing not just the sequences, but also various types of lab-generated 'linear' data relevant to genome interpretation -- for example read depth, or protein binding information.

Probability Models

- We use *probability models* for this purpose:
  - genomes are products of a probabilistic process (evolution)
  - detecting biological "signal" against "noise" of background sequence or mutations
  - measure of reliability

29

So, what *type* of model? It won't be a surprise that we favor probability models, for several reasons: First, genomes result from evolution, which is inherently probabilistic as we discussed earlier. Second, what we're trying to do is basically to detect site signals in a noisy background, and for signal to noise problems probability models are a widely used and powerful approach. Noise almost by its definition has to be modelled probabilistically. And as we've seen, there's a lot of variability in site sequences, so it makes sense to model that probabilistically as well.

Finally, probability models allow you to report a measure of confidence in your prediction, which is important when you're trying to persuade someone to commit experimental resources to confirming it.

Models: simplicity vs complexity

- "*All models are wrong; some models are useful.*" – George Box
- "*What is simple is always wrong. What is not is unusable.*" – Paul Valery
- "*Everything should be made as simple as possible, but not simpler.*" – Albert Einstein (?)

30

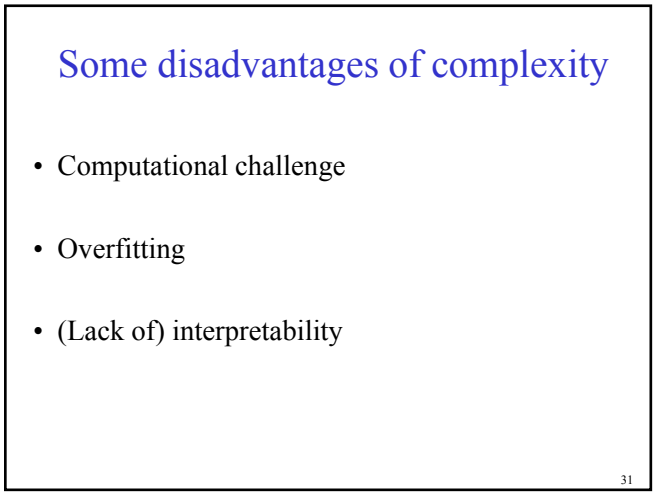
There is still an important and non-trivial question as to how complex our models need to be. To start with, here are a few illuminating quotes:

One, from the British statistician George Box, is 'all models are wrong; some models are useful'. He means that the real world is complex enough that no model can fully capture it, and so any model is 'wrong' in that sense. That's especially true of biological systems, which are far more complex than simple physical systems. But some models are able to capture enough of the reality to be able to make some progress in understanding it, and so are useful.

An older quote from the French poet and philosopher Paul Valery is 'What is simple is always wrong. What is not [by which he means what is not simple] is unusable'. He's talking more broadly about how we think about reality, and saying that we have no choice but to simplify.

An even earlier quote from Einstein is 'Everything should be made as simple as possible, but not simpler'. He's talking here about the process of trying to deduce physical laws, and is saying, again, that you have to simplify reality to some extent, but that it's important to simplify to the appropriate level.

Okay, so all models are going to be oversimplifications to some extent, but that still doesn't tell us how complex they should be. Generally, we start with simple models ('as simple as possible' as Einstein would say), and then gradually make them increasingly complex to capture more and more of the biology.



There's a lot of exciting research now in developing and applying deep neural nets, which are extremely complex computational models that can have literally trillions of parameters, to all sorts of things including molecular biology. So now that we have the ability to do that, what's the point of using anything simpler? Well, complex models have some significant disadvantages that it's important to be aware of. In increasing order of seriousness:

One, of course, is that they're more computationally challenging to work with.

A second is ***overfitting***. The more parameters your model has, the more it will tend to capture chance characteristics of the training data that you use for estimating the parameters, with the result that the trained model although fitting the training set very well, performs poorly on new data not in the training set. And of course it's the new data that you're most interested in. Neural net developers are well aware of this issue of course and have methods to try to avoid it, but those methods don't work perfectly. For example, it has been found that deep neural nets for image classification typically fail on images to which a small amount of pixel noise has been added -- such that a human has no problem seeing what's in the image, but the neural net breaks down because it's 'overfit' to data lacking such noise. 'Hallucinations' are probably another manifestation of overfitting.

Statisticians have given a lot of thought to the issue of overfitting in complex models, but I think it's fair to say it's not really a solved problem yet — you can reduce overfitting, but not eliminate it. Even simple models will tend to overfit to some extent.

Finally, and this I think is really the most serious issue, is that complex models are difficult to interpret: what do the trillions of parameters mean? How does the model really 'work'? Basically, a complex model is a 'black box'. You can run it on data, and get an answer, and it might even be a reliable answer, but you don't understand why you got that answer. I'd argue that this is in fact anti-scientific, since the goal of science is to understand reality and a black box that simply makes predictions doesn't provide that understanding. The machine learns, but we don't! Again, people who work on deep neural nets recognize model interpretability as an important issue, and have made some progress on it, but it's a very hard problem and I think it's fair to say that it is far from being solved.

This course

- The focus is ***sequence-based*** CMB
  - i.e. methods (& models) for obtaining & analyzing the information encoded in the genome
- We emphasize the underlying ***biology***
- ***Simple / interpretable*** computational models are favored
- ***Proofs*** are often only intuitive sketches, omitting details

32

In this course we'll focus on sequence-based computational molecular biology; that is, general methods for obtaining and analyzing the information encoded in the genome sequence. As I mentioned earlier, many of the methods apply more broadly to other types of linear data associated with genomes.

We'll emphasize the underlying biology, discussing along the way some of the biological facts that are relevant to the computational methods.

We'll favor simple and interpretable probability models. This is not to discount the importance of more complex machine learning models, but rather it's because if you're going to be working with probability models at all it's good to start with simple ones. And as we'll see you can get surprisingly far with those. I suspect there's a place for models that are intermediate in complexity between deep neural nets and the fairly simple hidden Markov models that we'll be talking about in this course -- still simple enough to be interpretable, but incorporating more of our knowledge of the genome. But that's for future research!

Regarding proofs: Typically I just try to give you some intuition for proofs, and either omit the details altogether or include them on slides that I skip over in the lecture. This reflects how I personally tend to absorb proofs which is to read through them several times, each time getting some of the ideas. That doesn't fit very well into a lecture. Hopefully you won't find that too frustrating.

Main topics

- ***Suffix arrays*** (& hash tables) for finding exact matches
- ***Background sequence models***
- ***Site models,*** weight matrices & sequence logos
- Highest weight paths on weighted directed acyclic graphs: ***dynamic programming algorithm***
- Finding non-background-like regions ("HMMs lite")
- Edit graphs & ***gapped-alignment algorithms***
- ***Hidden Markov models*** and applications
  - Parsing genomes (into sites & non-sites)
  - Finding conserved regions
- Simple molecular evolution models

33

Here's a survey of the course content; it is laid out in more detail on the course web page.

In the next lecture we'll discuss algorithm generalities and then suffix arrays and hash tables, which are methods for finding exact sequence matches.

Then, probability models for background sequence.

Then, probability models for sites, and related to those, weight matrices and sequence logos, which have to do

with comparing two models, namely site models and background models.

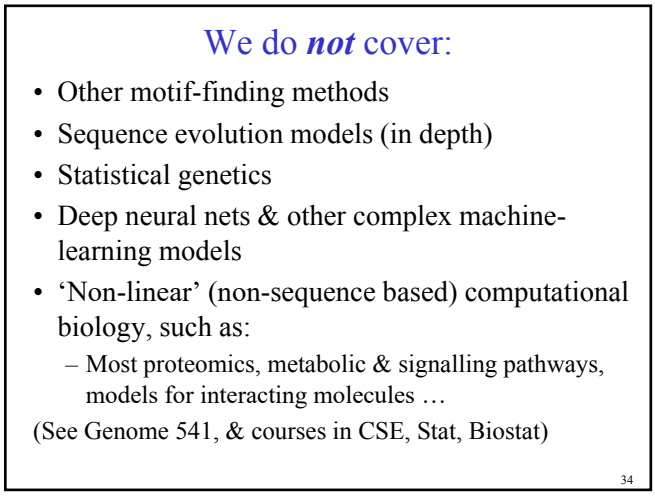
Then I want to talk about dynamic programming, which is sometimes called the "Fundamental algorithm of computational molecular biology" because it comes up in many different contexts. I'm going to present this algorithm in the context of finding highest weight paths on weighted directed acyclic graphs, for two reasons: First, I find it very helpful to have the concrete picture of a graph for understanding how the algorithm works. Second, in the applications we want to make of this algorithm in computational biology, there nearly always is an associated graph for which highest weight paths or related constructions correspond exactly to what you want to compute.

Then I'll discuss the problem of finding sequence regions which compositionally don't look like the background. I call this 'HMMs lite' because the ideas are closely related to what comes up in 2-state hidden Markov models.

Then I'll talk about gapped-alignment algorithms. The weighted directed acyclic graph that's associated to these is called an edit graph, and the Smith-Waterman algorithm is just dynamic programming to find a highest weight path on this graph. We'll talk about those more generally for aligning multiple sequences.

Then we'll get to hidden Markov models and some of their applications. In particular, HMMs are a good way of parsing genomes into sites and background. HMMs greatly generalize site models and background models, which you can think of as corresponding to different states or sets of states within a genomic HMM.

And finally finding conserved regions within alignments. PhastCons (which I showed you a slide from earlier) uses what's called a phyloHMM, so we'll talk about those. And in the context of that I'll talk about simple molecular evolution models, because you need those in phyloHMMs. There's a lot more to say about molecular evolution models that I won't get into. I'll just give you a simple idea of how to do some of the basic calculations.



Things we don't cover:

Although HMMs can be used to find new site motifs, there are other approaches that can be more powerful.

Deeper discussion of sequence evolution models.

We don't talk about statistical genetics, so we don't tell you how to do a GWAS, for example.

We don't talk about deep neural nets or other complex machine-learning models (beyond HMMs).

You could argue that all of the above are relevant in interpreting genomes, so they're part of 'linear' computational molecular biology. So we don't cover all of linear CMB. Nor do we cover 'non-linear' computational biology, which goes beyond genome sequences. For example, modelling the 3D structure of a protein, metabolic and signalling pathways, models for interacting molecules within a cell.

Many of these topics are covered in other UW courses. In Genome 541, the sequel to this course, the content varies from year-to-year, but in the past has often covered some of these topics; also courses in computer science, statistics and biostatistics.

That concludes this overview lecture. In the next lecture we'll discuss algorithms, and specifically a clever one called the suffix array algorithm which you'll need to implement in the first homework assignment.