Methods in Bioinformatics, 1-BIN-301/2-AIN-501

Lecturers:

Broňa Brejová, M-163, brejova@fmph.uniba.sk
Tomáš Vinař, M-163, vinar@fmph.uniba.sk

Web: http://compbio.fmph.uniba.sk/vyuka/mbi/

Announcements, Q&A: Microsoft Teams code h2w2fxb (guests please write e-mail)

Literature:

$\boxed{\text{I-INF-D-23}}$: Durbin, Eddy, Krogh, Mitchison: Biological sequence analysis. Cambridge University Press 1998.
$\boxed{\text{I-INF-Z-2}}$: Zvelebil, Baum: Understanding Bioinformatics. Taylor&Francis 2008.

Some lecture notes (in Slovak), notes and videos on the web page

Times and Lecture Rooms

- Lecture Thu 15:40-17:10 lecture hall C
- Tutorials (CS) Thu 14:00-15:30 lecture hall C
- Tutorials (Bio) Thu 17:20-18:50
  lecture hall C and computer room M-217

we plan to record/stream lectures and CS tutorials

“CS”: students of computer science, bioinformatics, applied informatics; please enrol under 1-BIN-301 code

“Bio”: students from the Faculty of Natural Sciences, students of biomedical physics; please enrol under 2-AIN-501 code

others: contact us

Course Goals

- **Everyone:** Overview of basic methods for analysis of biological sequences and other data sets in molecular biology
- **CS:** Algorithms and data structures, machine learning, probability. How to develop mathematical abstractions for real-world problems.
- **Bio:** Mathematical models at the core of popular bioinformatics tools, how to use tools, interpretation of their results.
- **Everyone:** Experience with an interdisciplinary collaboration.

Grading

3 homework assignments 30% (10% each)
Journal club 10%
Quizzes 10% (1 point each week)
Final exam 50%

(no quizzes for English speaking guests)

Final grade: A: 90+, B: 80+, C: 70+, D: 60+, E: 50+
At least 50% of the final exam is required

- Two versions of questions: bio and CS
- Journal club: read a research paper, write summary in a group (optional presentations for bonus points)
- You are allowed 2 double sided A4 pages as a cheat sheet on the exam
- DO NOT COPY, DO NOT CHEAT!

What to expect from lectures and tutorials

Typical lecture

- Biological introduction to a problem
- Formulation/abstraction as a computer science problem
- Algorithm idea for the problem solution(s)

Typical tutorial

- CS: algorithmic details and extensions, background biological knowledge
- Bio: applications to concrete data sets, what do various parameters mean and how to set them, background computer science knowledge,

Weekly quizzes

- Cca 5 short questions concerning last week's lectures and tutorials
- Due on Wednesday 10pm
- Moodle link on the web page
- Goal: review basic concepts from the lecture and the tutorial
- **First quiz already this week**

Example from our research

common marmoset, Callithrix jacchus, 250g, 18cm

Genome sequenced in 2007

(Washington University St. Louis a Baylor College of Medicine, USA)

Analysis published in 2014

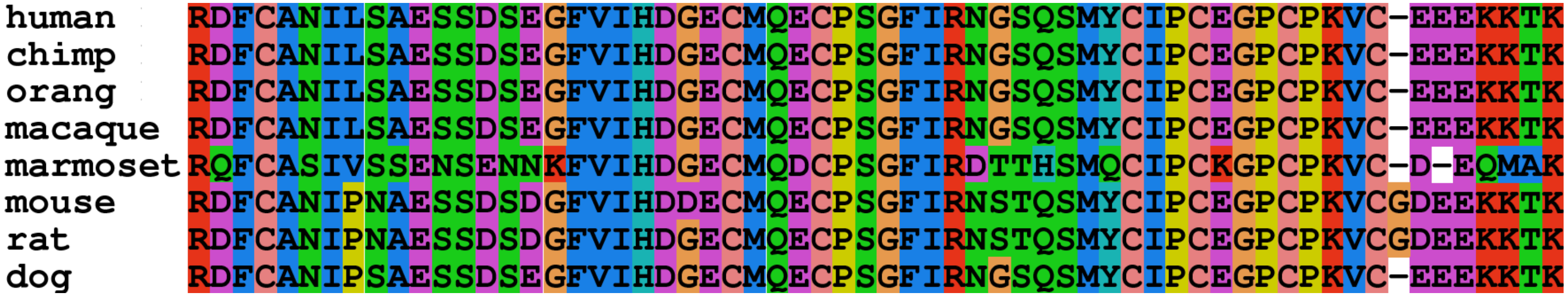
IGF1R: Insulin-like growth factor 1 receptor

Protein passes through cytoplasmic membrane on the cell surface

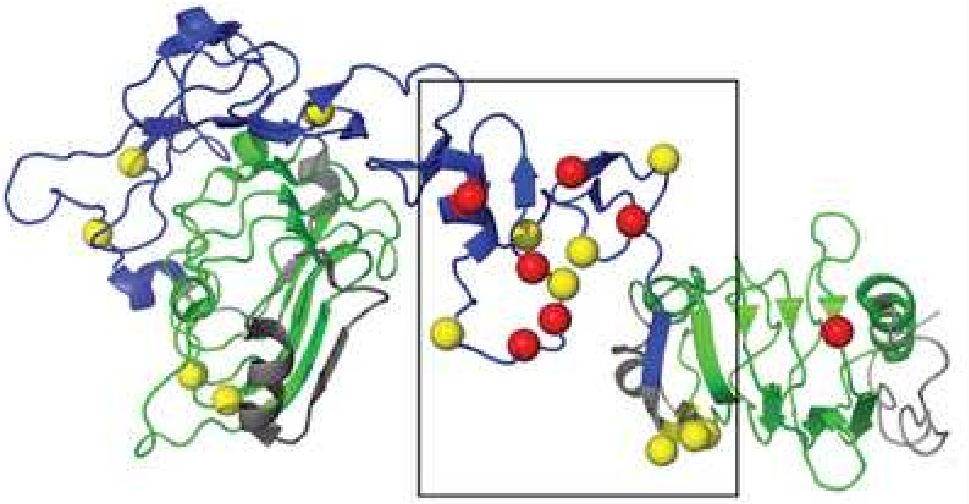
After binding to growth hormones IGF1, IGF2 signals into the cell

Functions related to the cell growth and division,

organism growth, cancer



```
human    RDFCANILSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
chimp    RDFCANILSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
orang    RDFCANILSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
macaque  RDFCANILSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
marmosetRQFCASIVSSENSENNKFVIHDGECMQDCPSGFIRDTTHSMQCIPCKGPCPKVC-D-EQMAK
mouse    RDFCANIPNAESSDSDGFVIHDDECMQECPSGFIRNSTQSMYCIPCEGPCPKVCGDEEKKTK
rat      RDFCANIPNAESSDSDGFVIHDGECMQECPSGFIRNSTQSMYCIPCEGPCPKVCGDEEKKTK
dog      RDFCANIPSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
```

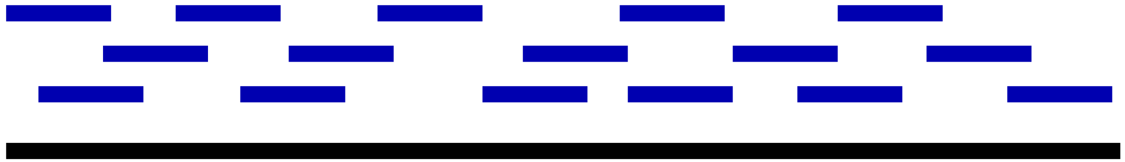


What bioinformatics tools were needed for this research?

1. Assemble genome from sequencing reads
2. Find sequence similarities to other genomes
3. Find genes coding for proteins
4. Find genes under positive selection
5. Determine structure and function of the proteins

1. Genome assembly

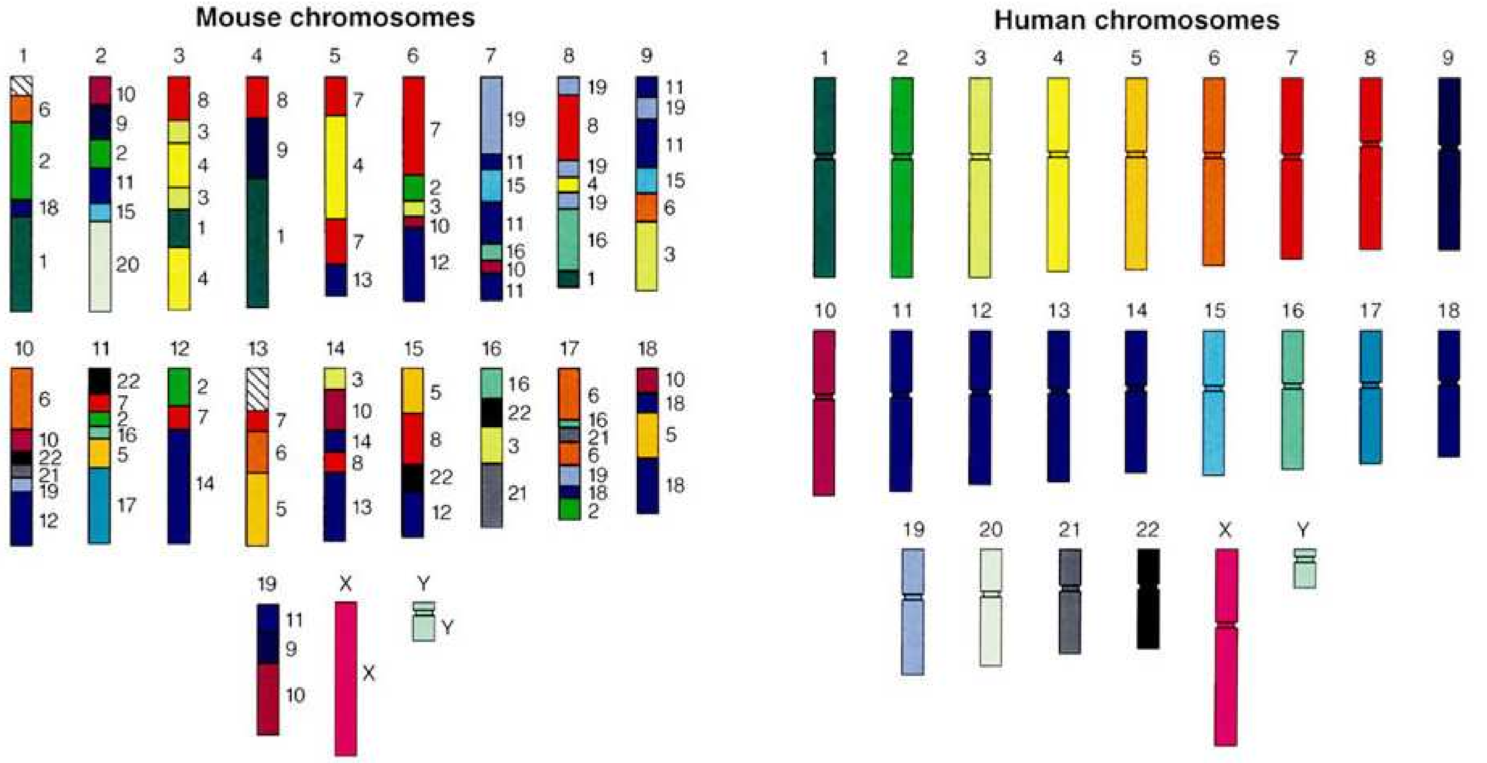
- We can only sequence short fragments of DNA (e.g. of length 1000)
- Each place in the genome is sequence multiple times (for marmoset on average 6×)



- We need to "glue" sequencing reads together based on overlaps
- Huge amount of data ⇒ need efficient algorithms

2. Finding similarities to other genomes

For each place in the marmoset genome find corresponding places in other genomes (e.g. human, chimp, mouse, . . .)

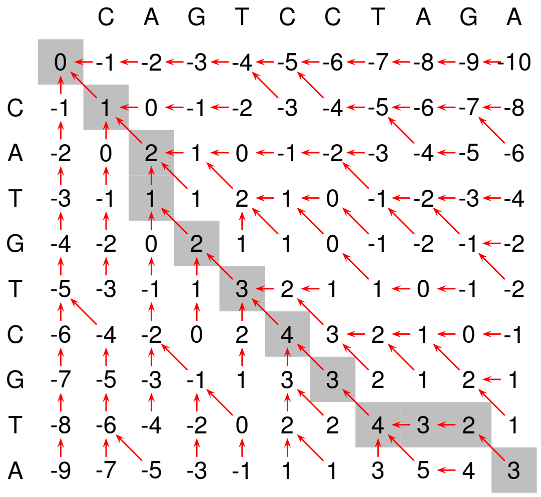


2. Finding similarities to other genomes

- We are looking for similarities between DNA sequences

```
  Human AGTGGCTGCCAGGCTG---GGATGCTGAGGCCTTGTTTGCAGGGA
 Rhesus AGTGGCTGCCAGGCTG---GGTTGCTGAGGCCTTGTTTGCCGGGA
  Mouse GGTGGCTGCCGGGCTG---GGTGGCTGAGGCCTTGTTGGTGGGGT
    Dog AGTGGCTGCCCGGCTG---GGTGGCTGAGGCCTTATTTGCAGGGA
Chicken AGTGGCTGCCAGTCTGCGCCGTGGCCGACGTCTTGCTCGGGGGAA
```

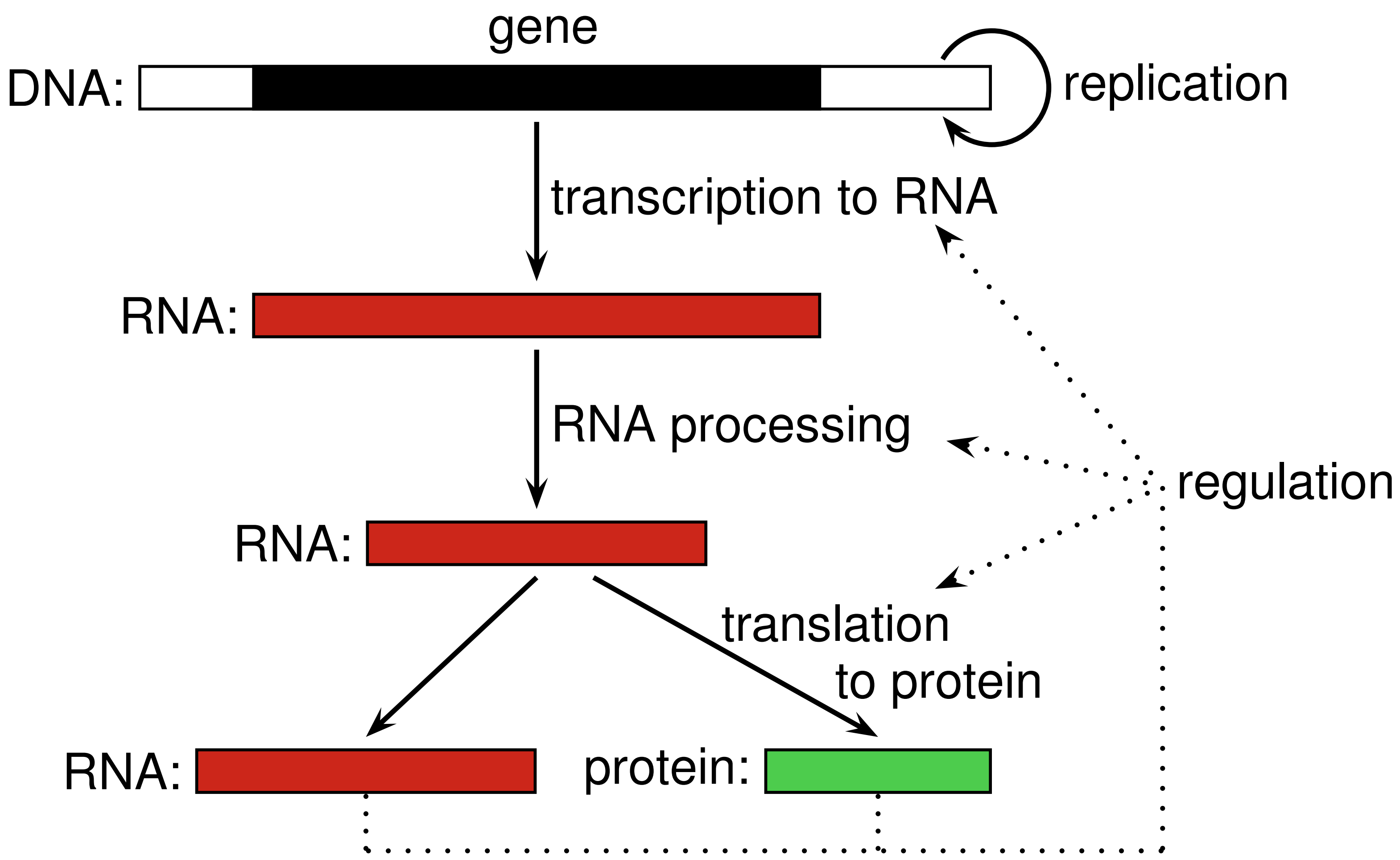
- Basic technique used here is called **dynamic programming** which can decompose a large problem into many smaller (and easier) ones



		C	A	G	T	C	C	T	A	G	A
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
C	-1	1	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-2	0	2	1	0	-1	-2	-3	-4	-5	-6
T	-3	-1	1	1	2	1	0	-1	-2	-3	-4
G	-4	-2	0	2	1	1	0	-1	-2	-1	-2
T	-5	-3	-1	1	3	2	1	1	0	-1	-2
C	-6	-4	-2	0	2	4	3	2	1	0	-1
G	-7	-5	-3	-1	1	3	3	2	1	2	1
T	-8	-6	-4	-2	0	2	2	4	3	2	1
A	-9	-7	-5	-3	-1	1	1	3	5	4	3

- The table is very large, in practice many improvements and heuristics to make this practical

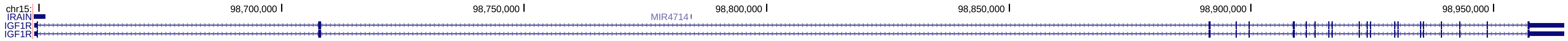
3. Finding genes coding for proteins



Which parts of the sequence genome code for proteins

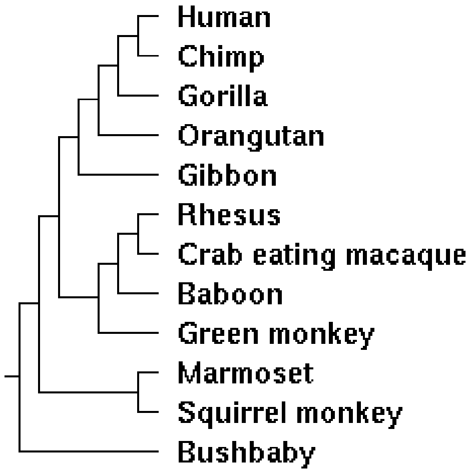
3. Finding genes coding for proteins

- Needle in a haystack: only 1% of human genome codes for proteins
- Code for a single protein is broken into many short parts (exons)
- IGF1R covers 315 569nt, but only 4101nt in 21 exons code for the protein



- Take known genes, collect various statistics
  find other regions of the genome with a similar statistical profile

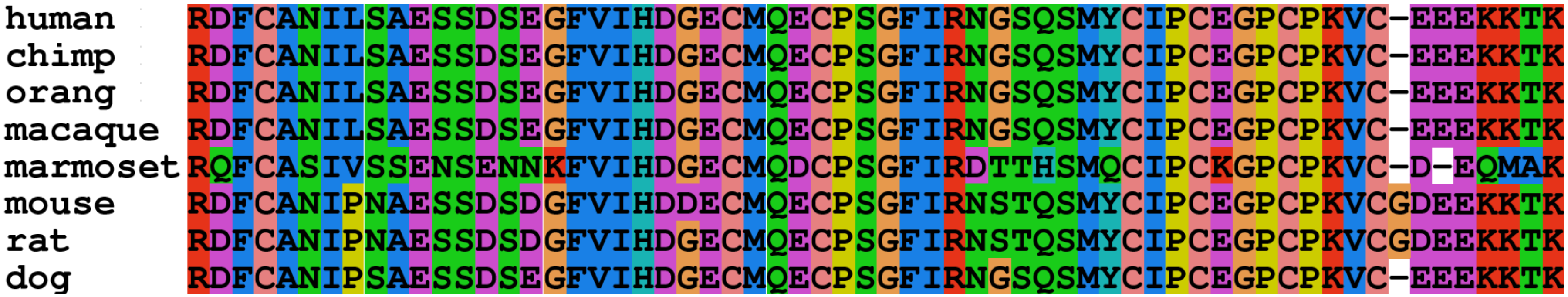
4. Search for genes under positive selection



- Study of evolutionary processes
- Mutations in DNA over time are subject to natural selection
- Most of random changes in a protein are harmful, thus segments encoding proteins typically mutate very slowly

4. Search for genes under positive selection

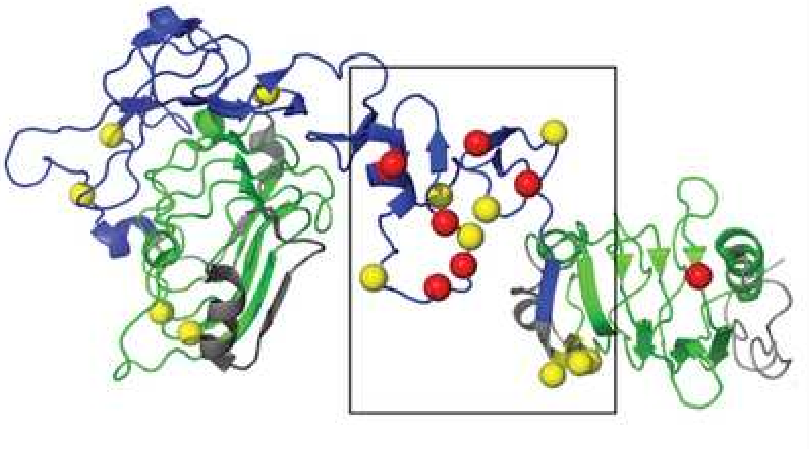
- Sometimes a beneficial mutation is discovered, followed by a surge of other mutations optimizing the new function → positive selection



```
human    RDFCANILSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
chimp    RDFCANILSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
orang    RDFCANILSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
macaque  RDFCANILSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
marmoset RQFCASIVSSENSENNKFVIHDGECMQDCPSGFIRDTTHSMQCIPCKGPCPKVC-D-EQMAK
mouse    RDFCANIPNAESSDSDGFVIHDDECMQECPSGFIRNSTQSMYCIPCEGPCPKVCGDEEKKTK
rat      RDFCANIPNAESSDSDGFVIHDGECMQECPSGFIRNSTQSMYCIPCEGPCPKVCGDEEKKTK
dog      RDFCANIPSAESSDSEGFVIHDGECMQECPSGFIRNGSQSMYCIPCEGPCPKVC-EEEKKTK
```

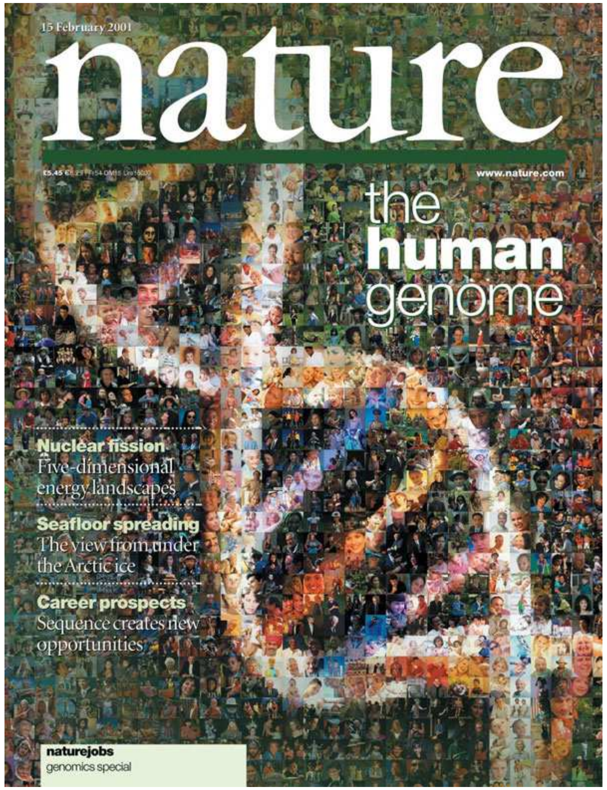
5. Determining function and structure of proteins

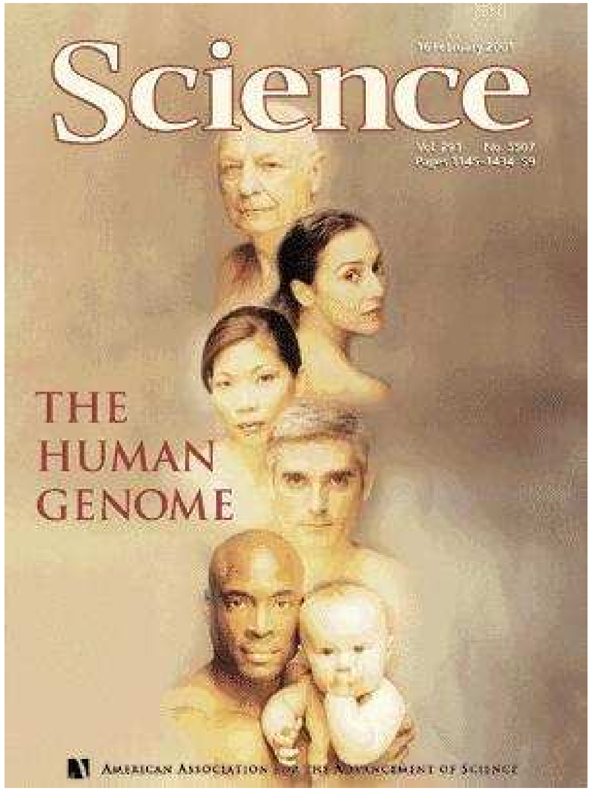
- After steps 1-4, we have a list of 37 genes under positive selection in the marmoset genome
- What is their function? Any of them related to marmoset size?
- What is the shape of the protein, where are the position under positive selection located?
- Protein structure (shape) can be determined experimentally expensive and time consuming, instead 3D structure predictions



Genome Sequencing and Assembly
(Sekvenovanie a zostavovanie genómov)

Tomáš Vinař
23.9.2021





DNA Sequencing Overview

1. Chromosomes are cut randomly into smaller fragments
   (e.g. using **sonication**)

2. Each fragment is copied multiple times
   (e.g. through PCR, bacterial cloning, . . .)

3. Ends of fragments are sequenced by one of the sequencing technologies
   $\Rightarrow$ many short strings called **reads**

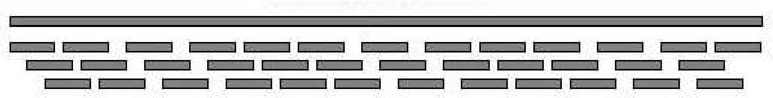
4. Short strings are **computationally assembled** back into chromosomes

Overview of Sequencing Technologies

Technology	Read length	Errors	Output per day	Cost per MB
1st generation				
Sanger	up to 1000bp	$< 1\%$	3 MB	\$4000
2nd (next) generation (cca 2004)				
Illumina	250bp	$< 0.1\%$	150 GB	\$0.03
3rd generation (emerging)				
PacBio	cca 14kbp	10%	700 GB	\$0.02
PacBio HiFi	cca 15kbp	$< 1\%$	70 GB	\$0.20
Oxford Nanopore	really long	up to 10%	50 GB	\$0.02

Bioinformatics Problem: Sequence Assembly (zostavenie genómu)

- **Input:** short DNA fragments (reads)
- **Goal:** reconstruct the sequenced genome
  — using sequence identity in overlapping reads
- Important factors:
  - **Size of the genome**
  - **Length of individual reads**
  - **Coverage** — how many times on average is the genome covered?



Simple but Unrealistic Formulation

Shortest common superstring problem.

We are given several strings $S_1, \ldots S_k$ (sequenced reads), find the shortest string S containing each S_i as a (contiguous) substring

Motivation: use overlaps between reads as much as possible

Example:

Input: GCCAAC,CCTGCC,ACCTTC
Output: CCTGCCAACCTTC (reads connected in order S_2, S_1, S_3)

Shortest Common Superstring

- **NP hard problem**
  no known polynomial-time algorithm can find optimal answer for each input
- **Simple heuristics:** repeatedly find two reads with longest overlap and connect them to a single read
- Example: CATATAT, TATATA, ATATATC
  Optimum: CATATATATC, length 10
  Heuristics: CATATATCTATATA, length 14
- This heuristics is an **approximation algorithm:**
  It finds a string which is at most $3.5\times$ longer than optimal superstring
- Conjecture: it is in fact a 2-approximation algorithm
- There is a different 2.5-approximation algorithm

Shortest Common Superstring: Unaccounted Factors

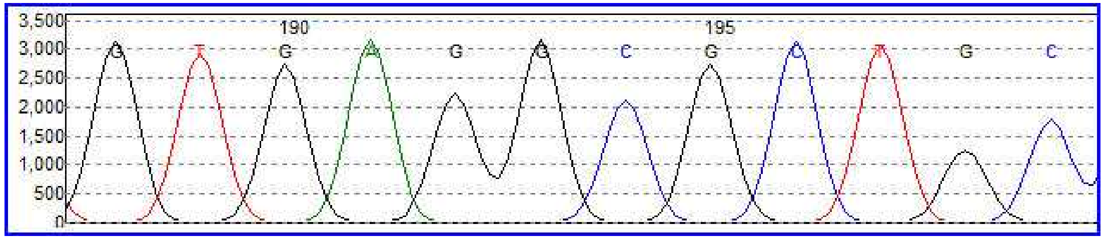
- Sequencing errors
- Polymorphism
- Two strands (reads come in two different orientations)
- Contamination (e.g. by DNA from bacteria used for cloning), chimeric reads
- Multiple chromosomes, incomplete genome coverage
- Sequence repeats
  cca 50% of human genome is repetitive DNA
  Example: 10xTTAATA, 10xATATTA, 3xTTAGCT
  TTAATATTAGCT?
  TTAATATTAATATTAATATTAATATTAGCT?
  TTAATATTA + ATATTAGCT?

Unaccounted factors: base quality

- Reads typically accompanied by **base qualities**
  How likely is this base correct?

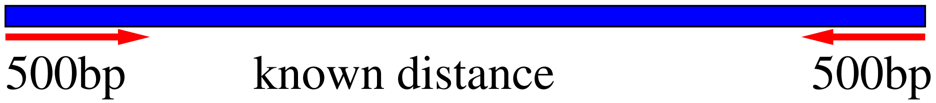
- Base with quality $q \Rightarrow$ probability of error $10^{-q/10}$
  i.e. base with $q > 40$ is correct for 99.99%

Example of Sanger sequencing result (trace):



Shortest Common Superstring: Simplifying Factors

Additional information: pair-end reads



plasmid 2–10 kB
cosmid 40 kB

Simplification: we do not need to connect everything to one string, we connect only parts bridged by multiple reads.

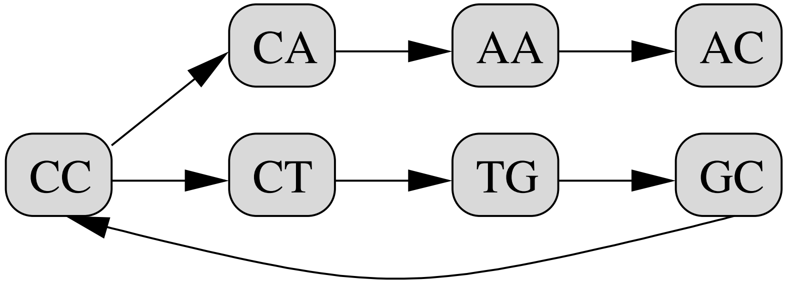
Conservative approach: sacrifice completeness for accuracy

Shortest Common Superstring: Summary

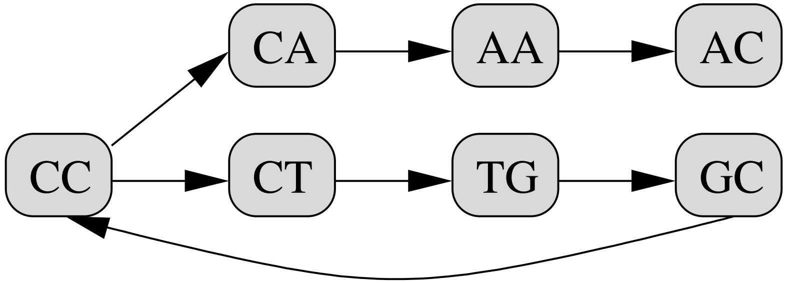
- Unrealistic formulation and difficult problem
- Perhaps theoretical problem can yield some insights into real application?
- Overlap-Layout-Consensus approach motivated by greedy algorithms (join fragments with large overlaps)

Assembling Short Reads: de Bruijn Graphs

- Split reads to overlapping windows of length k
- **de Bruijn graph** of dimension k is a **directed graph**:
  - **vertices:** substrings of length k from all reads
  - **directed edges:** connect k-mers consecutive in at least one of the reads (overlapping by $k-1$ bases)
- **Example:** $k = 2$, reads: CCTGCC, GCCAAC



How to use de Bruijn graph for assembly?



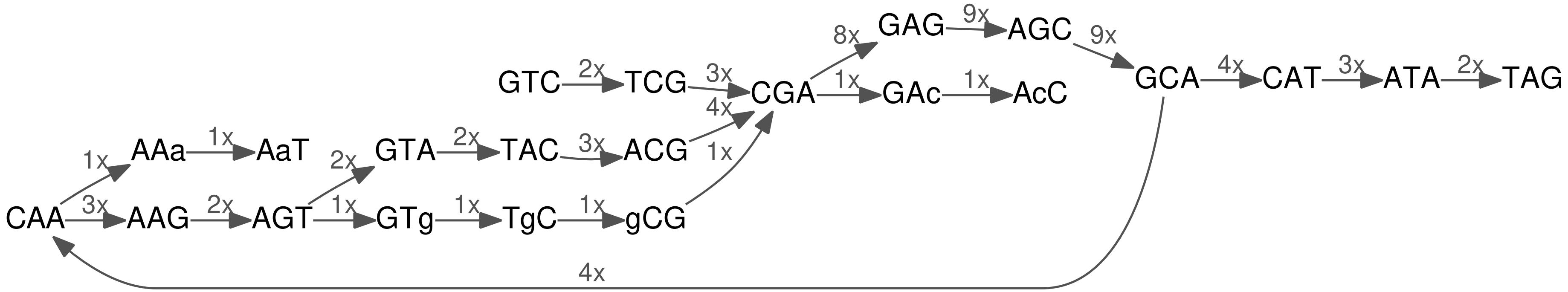
- If there was only a single chromosome and there were no ambiguous k-mers, the correct assembly would correspond to a **Eulerian path:** a path in the graph which uses each edge exactly once

- We can easily test if such a path exists and to find it in $O(m + n)$

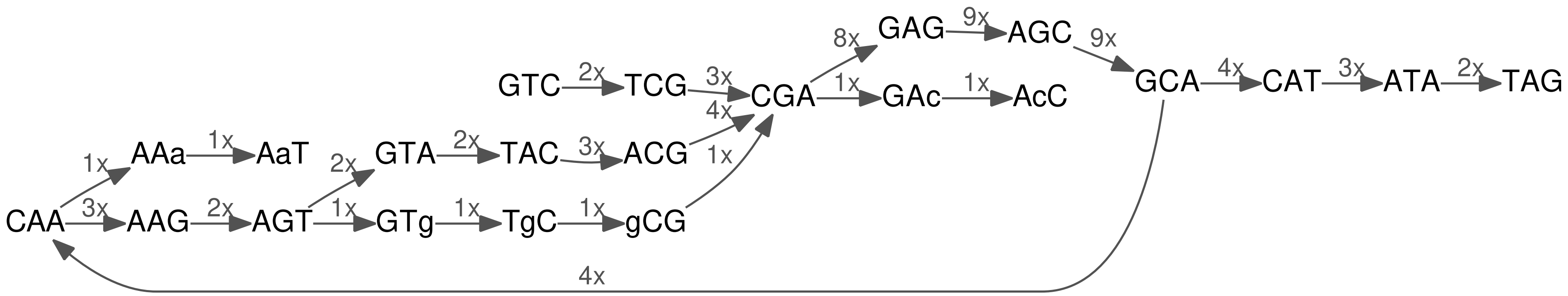
- In general, assembly will correspond to a set of **walks in the de Bruijn graph** covering most edges

Example: reads and their de Bruijn graph

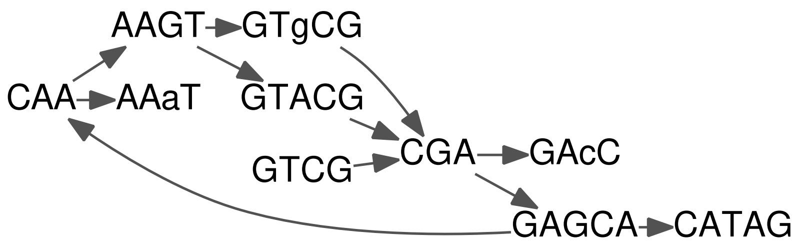
```
GTCGAGCAAGTACGAGCATAG
 TCGAGCA      AGCATAG
    AGCAAaT   AGCATAG
GTCGAcC  GTACGAG
GTCGAGC   TACGAGC
  CGAGCAA  ACGAGCA
        AGTgCGA
      CAAGTAC
     GCAAGTA GAGCAT
   GAGCAAG   GAGCATA
          TACGAGC
```



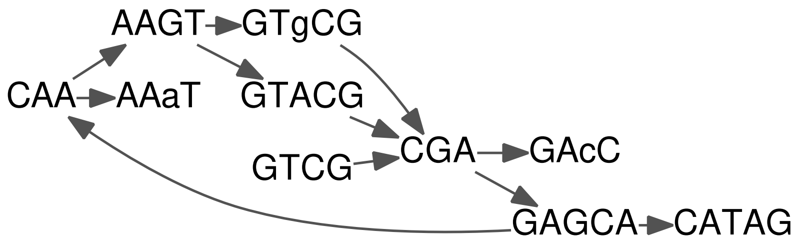
Example: simplifying de Bruijn graph



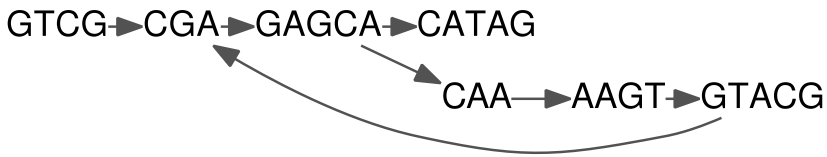
Unique paths are contracted to a single vertex



Example: removing errors from de Bruijn graph

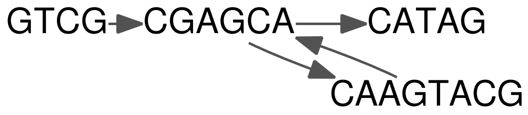


Remove tips and bubbles with low coverage



Contract unique paths again ⇒ four **contigs**

(originally $\texttt{GT}\underline{\texttt{CGAGCA}}\texttt{AGTA}\underline{\texttt{CGAGCA}}\texttt{TAG}$)



Typical Results of Assembly

- Many **short contigs** that can be further combined to **longer scaffolds** by using pair-end read information
- Some portions cannot be resolved due to **long repetitive sequences**

Example: Human chromosome 14, 88 Mbp, $70\times$ coverage (source: GAGE)

Method	Contigs	Errors	N50 corr
Velvet (basic de Bruijn)	>45000	4910	2.1 kbp
Velvet (with scaffolding)	3565	9156	27 kbp
AllPaths-LG	225	45	4.7 Mbp

N50: contigs with this length or longer contain 50% of the genome here N50 after error correction is shown

Summary

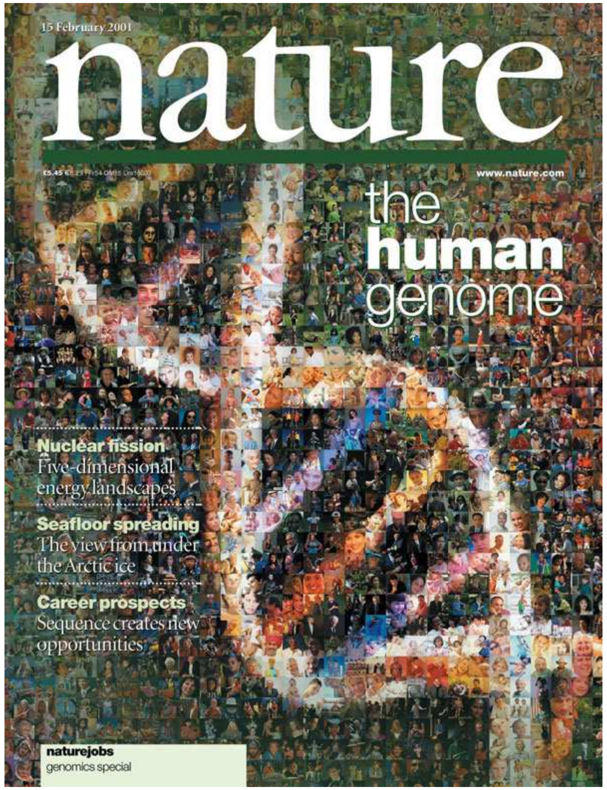
- Sequencing is a complicated process in which bioinformatics plays an important role
- Illumina technology offers extremely low price but only short reads
- Problem of genome assembly, shortest common superstring
- de Bruijn graphs: a practical solution for short reads
- Assembled sequence may contain errors, gaps, multiple contigs
- Next lecture: How to deal with 3rd generation reads?
- Genome coverage and read size are determining factors in how fragmented assembly will be:
  - for Sanger reads: typically $7-10\times$ coverage
  - for NGS reads: typically $40-70\times$ coverage
  - for 3rd generation: $30\times$ coverage

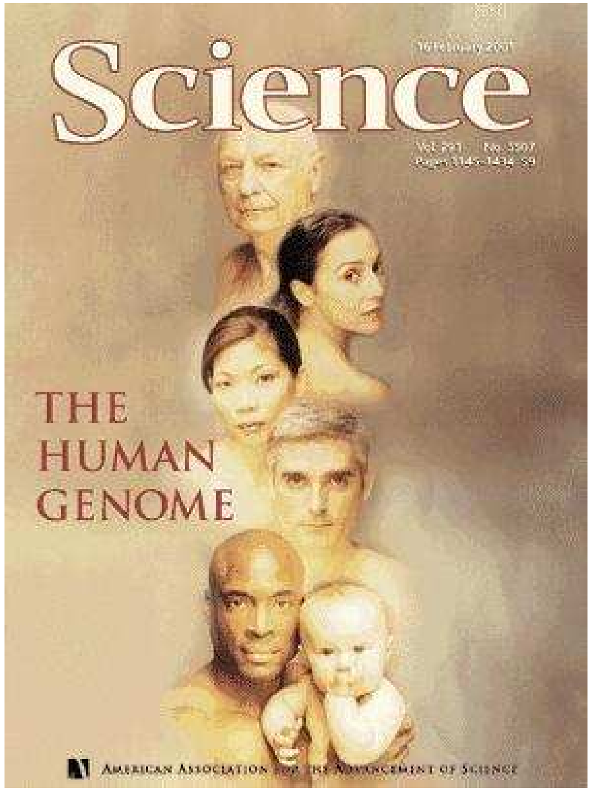
Genome Sequencing Milestones

1976	MS2 (RNA virus) 40 kB
1988	Human genome sequencing project (15 years)
1995	bacterium H. influenzae 2 MB, shotgun (TIGR)
1996	S. cerevisiae 10 MB, BAC-by-BAC (Belgium, UK)
1998	C. elegans 100 MB, BAC-by-BAC (Wellcome Trust)
1998	Celera: human genome in three years!
2000	D. melanogaster 180 MB, shotgun (Celera, Berkeley)
2001	2x human genome 3 GB (NIH, Celera)
after 2001	mouse, rat, chicken, chimpanzee, dog,. . .
2007	Genomes of Watson and Venter (454)
2012	1000 human genomes
soon	10k vertebrate genomes, sequencing as a diagnostic tool
2021	3.5 million SARS-CoV-2 genomes

Sequencing and Genome Assembly (part 2 - long reads)

Tomáš Vinař
30.09.2021





Overview of Sequencing Technologies

Technology	Read length	Errors	Output per day	Cost per MB
1st generation				
Sanger	up to 1000bp	$< 1\%$	3 MB	$4000
2nd (next) generation (cca 2004)				
Illumina	250bp	$< 0.1\%$	150 GB	$0.03
3rd generation (emerging)				
PacBio	cca 14kbp	10%	700 GB	$0.02
PacBio HiFi	cca 15kbp	$< 1\%$	70 GB	$0.20
Oxford Nanopore	really long	up to 10%	50 GB	$0.02

From the last lecture

- Genome is assembled from sequencing reads
- Genome assembly using de Bruijn graphs
- de Bruijn graphs not suitable for long reads with high error rate
  - "Disassembly" to k-mers throws away too much information (read length 10000+, k is usually between 30 and 70)
  - Error rate around 10% makes de Bruijn graph unwieldy (for $k = 31$, k-mer 3 errors on average)

Overlap–Layout–Consensus approach

- **Overlap:** Find overlaps between reads and create an **overlap graph**
- **Layout:** Simplify the overlap graph and find paths which will correspond to **contigs**
- **Consensus:** For each contig locate overlapping reads and construct a sequence as a consensus at each position (corrects local errors)

Overlap: Finding read overalps

```
CATCTCTAGGCCAGC
      |||||| ||
      TAGGCCTGCTTCTTG
```

- special case of the sequence alignment (next lecture)
- overlaps **will contain errors**
  (in our case approx. 1 error per 10bp of the overlap)
- **there are many reads:** $30\times$ human genome coverage
  $\Rightarrow$ cca 9 mil. of reads of length 10000
  we cannot afford to compare all pairs of reads
- practical approac:
  - fast pre-filtering of **suitable candidate pairs of reads**
    (for example those containing a common k-mer)
  - followed by a slower alignment for candidate pairs

Layout: Creating the overlap graph

- Example result from the previous phase:
  CATCTCTAGGCCAGC / TAGGCCTGCTTCTTG, overlap 9 bp
  . . .
- Create **overlap graph**:
  vertices: reads   weighted edges: overlaps and lengths

Example:

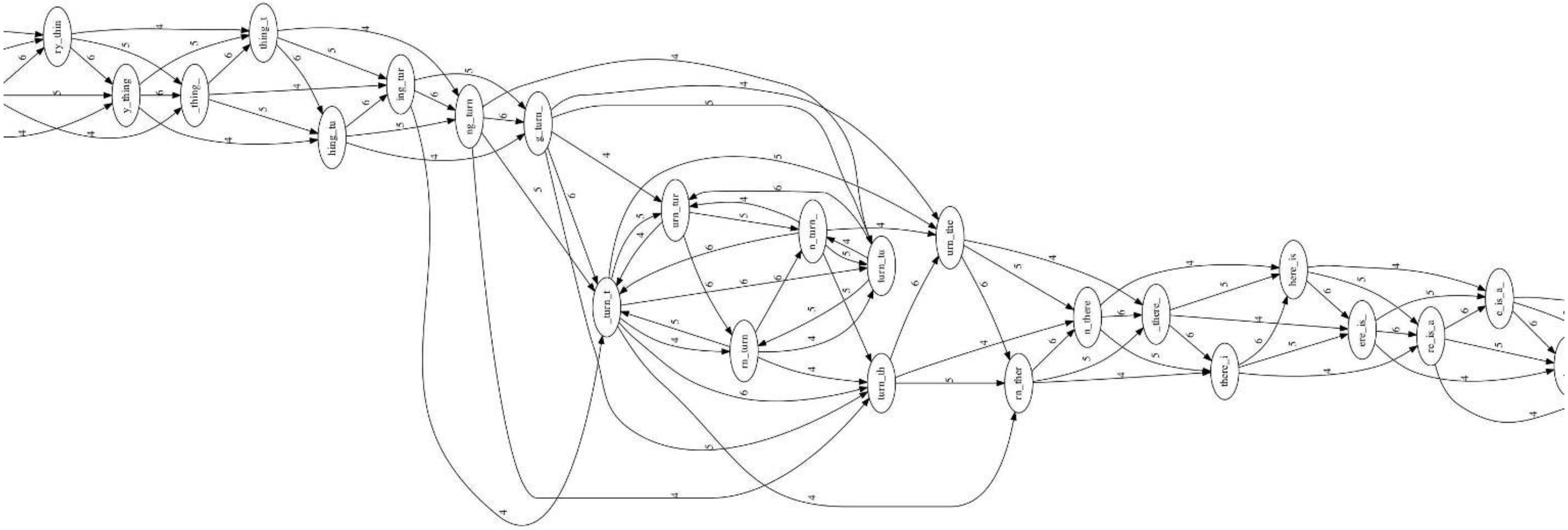
`to_every_thing_turn_turn_turn_there_is_a_season`

read length 7, minimum required overlap 4

Example:

`to_every_thing_turn_turn_turn_there_is_a_season`

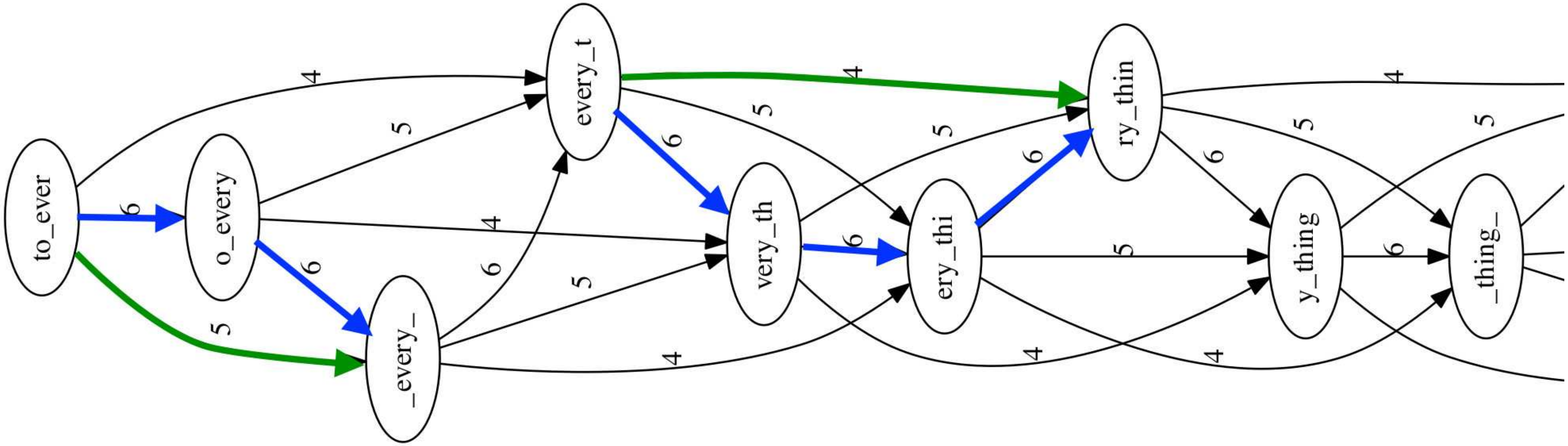
read length 7, minimum required overlap 4



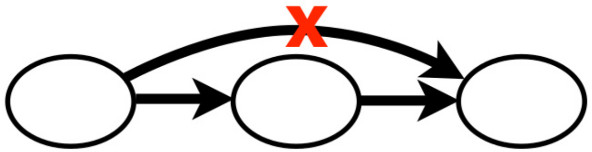
Example and figures by Ben Langmead

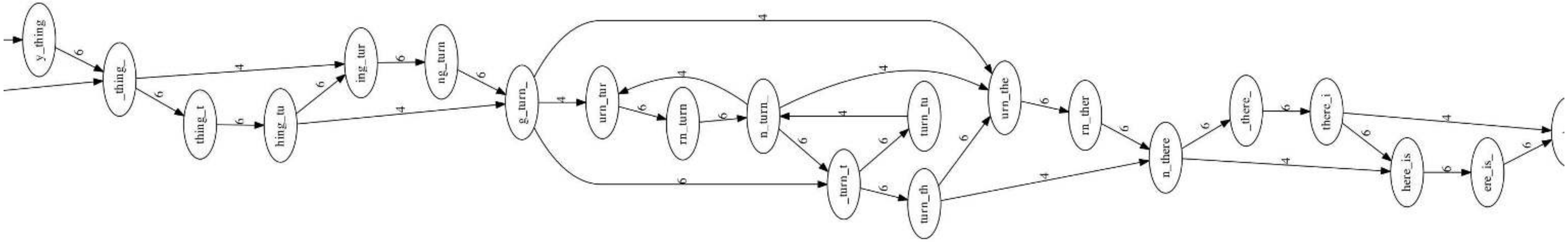
Layout: Transitive edges

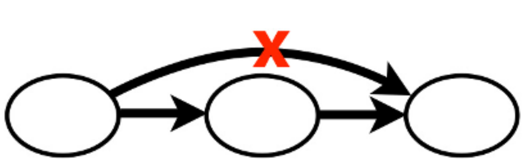
- Some edges are superflous because they say the same thing as other edges

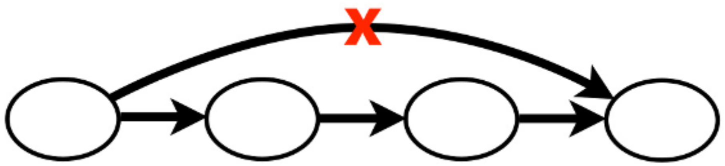


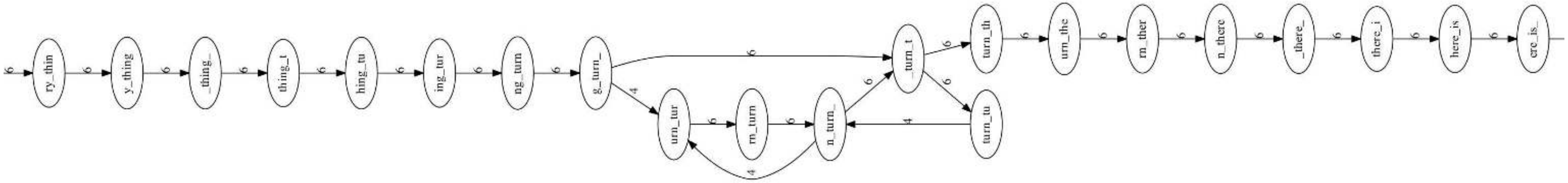
Layout: Removal of transitive edges









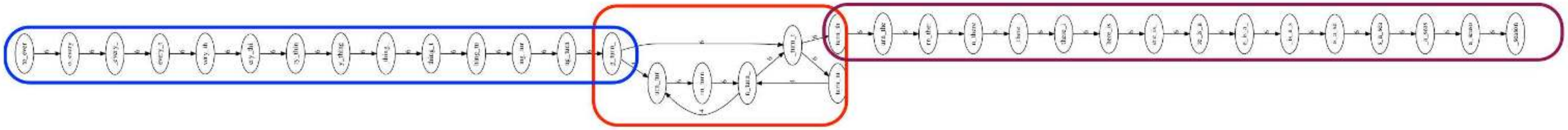


Layout: Identifying contigs

Original sequence:

`to_every_thing_turn_turn_turn_there_is_a_season`

Non-branching paths represent contigs



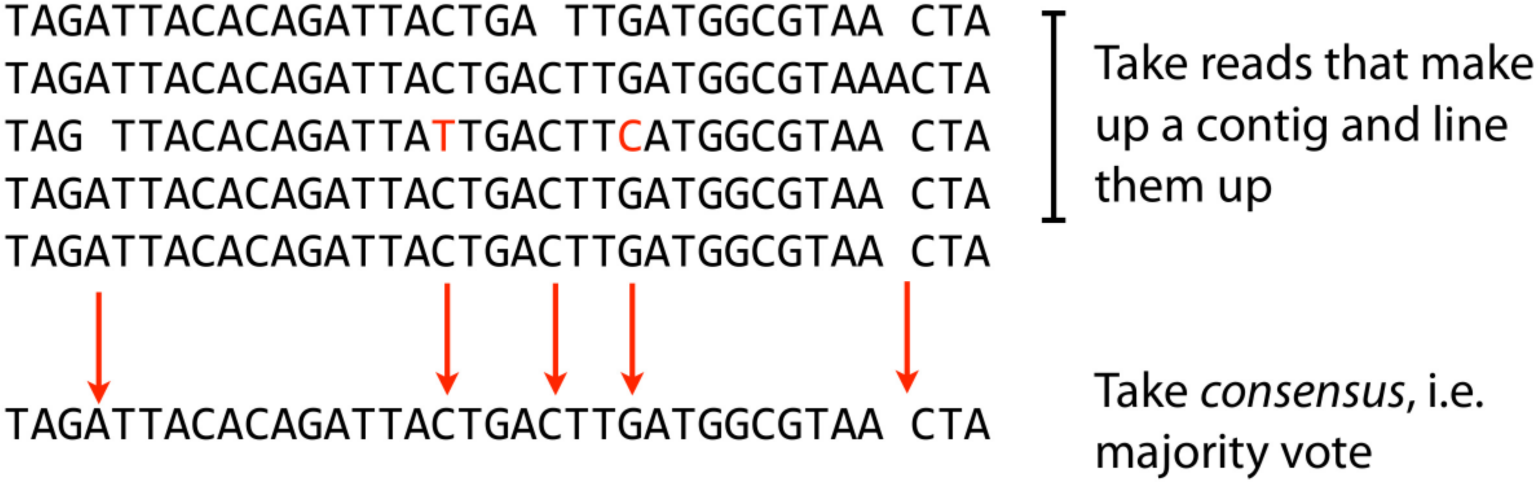
Result:

Contig 1
to_every_thing_turn_

Contig 2
turn_there_is_a_season

Unresolvable repeat

Consensus: Obtaining the final sequence



```
TAGATTACACAGATTACTGA TTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAAACTA
TAG TTACACAGATTATTGACTTCATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA

TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
```

Take reads that make up a contig and line them up

Take *consensus*, i.e. majority vote

Differences between de Bruijn graphs and the overlap graph

de Bruijn graphs

- fixed length of overlaps
- throw away information about contiguity spanning more than k bp
- genome represented by paths
- errors: bubbles and tips
- errors resolved in pre-processing
- contigs cover almost all edges

Overlap graphs

- variable length of overlaps
- use most of the information derived from overlaps
- genome represented by paths
- errors are “hidden”
- errors resolved in post-processing (consensus)
- transitive edges need to be removed

Example: Assembling genome of *Magnusiomyces capitatus*

(genome length 19.6 Mbp, 4 chromosomes + mtDNA)

Technology	Coverage	# contigs	largest	avg	N50
Illumina / Spades	250x	1102	172.6 Kbp	17.6 Kbp	62.0 Kbp
PacBio / Canu	37x	17	4.7 Mbp	1.2 Mbp	1.7 Mbp
PacBio + MinION	65x	11	4.4 Mbp	1.8 Mbp	2.0 Mbp

Summary

- Long reads allow us to assemble much more contiguous genome sequences compared to short reads
- Fast algorithms required to locate read overlaps (more in the next lecture)
- Overlap graphs and de Bruijn graphs are similar concepts attempts at unifying the two

Genome Sequencing Milestones

1976	MS2 (RNA virus) 40 kB
1988	Human genome sequencing project (15 years)
1995	bacterium H. influenzae 2 MB, shotgun (TIGR)
1996	S. cerevisiae 10 MB, BAC-by-BAC (Belgium, UK)
1998	C. elegans 100 MB, BAC-by-BAC (Wellcome Trust)
1998	Celera: human genome in three years!
2000	D. melanogaster 180 MB, shotgun (Celera, Berkeley)
2001	2x human genome 3 GB (NIH, Celera)
after 2001	mouse, rat, chicken, chimpanzee, dog,. . .
2007	Genomes of Watson and Venter (454)
2012	1000 human genomes
soon	10k vertebrate genomes, sequencing as a diagnostic tool
2021	3.5 million SARS-CoV-2 genomes

Use of NGS: Population genetics

- Obtain sequence reads from one individual
- What are the differences of the individual from the “reference” genome?
- How do genetic change influence phenotype?
- Personalized medicine
- Population structure and history
- Ethical questions

Bioinformatics problems:

- Mapping short reads to reference sequence
- Identification of differences (both local and large-scale)

Use of NGS: Environmental sequencing – metagenomics

- What microorganisms live in our bodies?
  gut flora, mouth, skin, . . .
- Microbial diversity in different ecosystems
- It is difficult to isolate individual species
- We can sequence a mixture of different genomes
- Then we try to assemble at least short contigs

Bioinformatics problems:

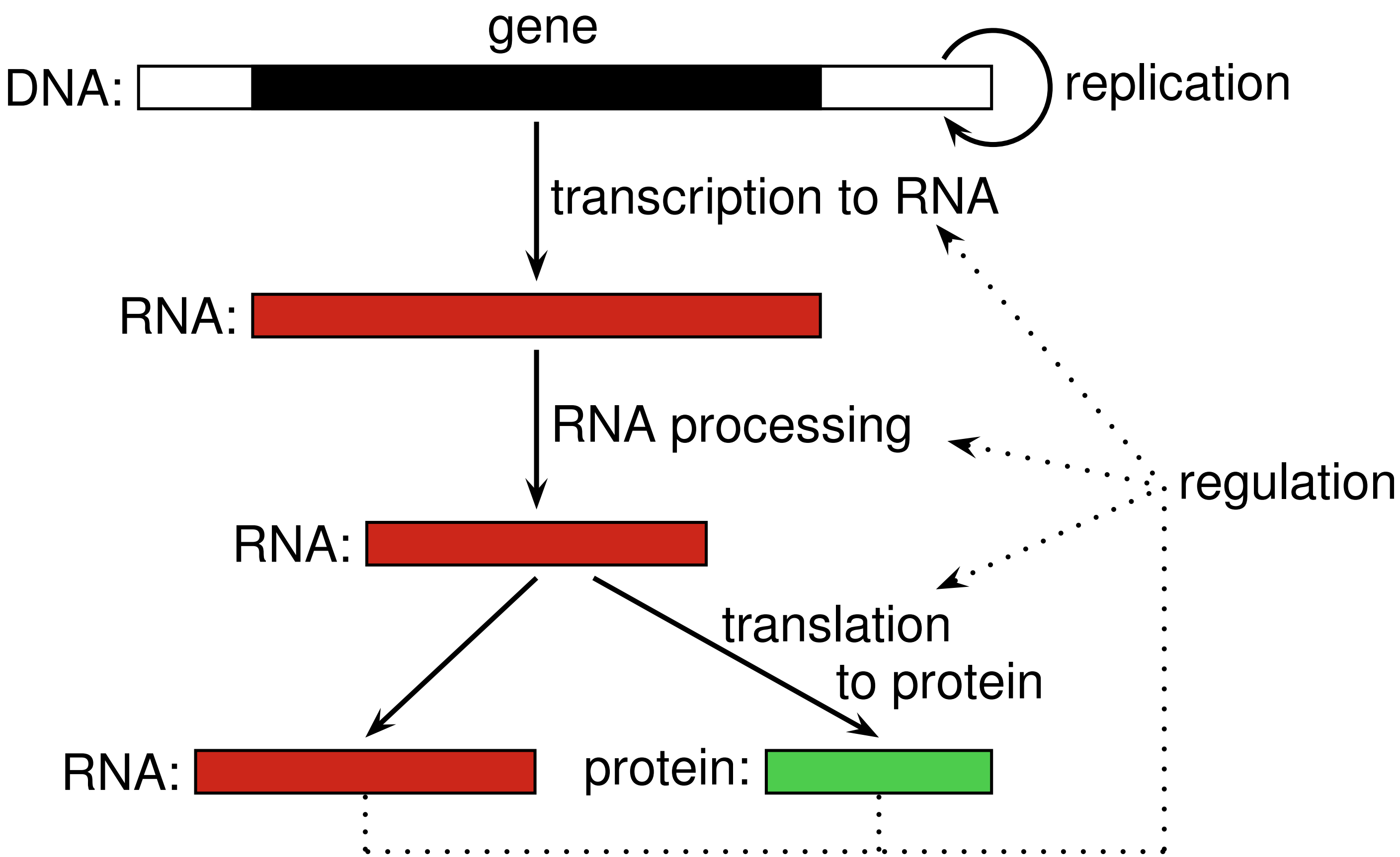
- Binning: Separation of reads from different genomes

Use of NGS: identification of genes, binding sites,. . .

- RNA-seq: sequencing mRNAs, obtaining positions of genes and their expression levels
- Chip-Seq: filtering DNA bound by a certain protein, sequening them and mapping to the genome

Bioinformatics problems:

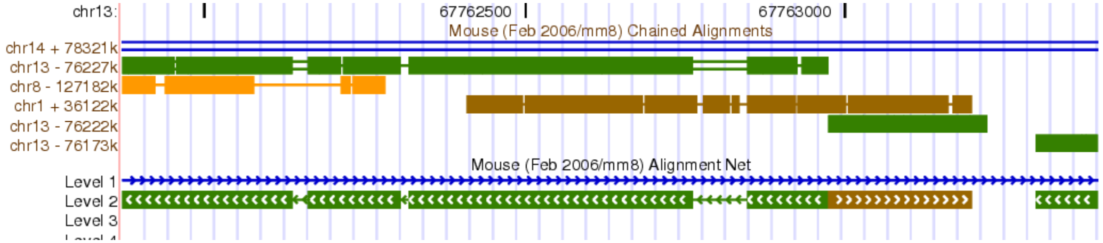
- Identification of splice sites
- Identification of binding sites using read coverage


gene
DNA:
replication
transcription to RNA
RNA:
RNA processing
regulation
RNA:
translation
to protein
RNA:
protein:

Sequence Alignment (zarovnávanie sekvencií) 1/2

Tomáš Vinař

October 7, 2021



[Durbin et al., 1998, chapter 2]

Problem: Local alignment

Input: two sequences

Problem: Local alignment

Output: similar regions (in the form of an alignment)

```
CCCGACGAGAAGGCCATAATGACCTATGTGTCCAGCTTCTACCATGCCTTT
|| ||||||||| ||||    ||||| |||  || || ||| ||   ||||
CCGGACGAGAAGTCCAT---CACCTACGTGGTCACCTACTATCACTACTTT
```

Insert dashes (gaps) so that corresponding bases in the same column.

A good alignment has many aligned matching bases, few gaps.

What are alignments good for?

- **Orientation in large sequence databases.**
  Genbank has more 3 TB of whole genome sequences.
  E.g.: from which genome (and which part) comes a given sequence?

- **Determine function (e.g. of a protein).**
  Similar sequences often have the same or similar function.

- **Evolutionary studies.**
  Search for homologs, sequences which have evolved from the same common ancestor.
  In the ideal case, gaps correspond to insertions and deletions, aligned bases to conserved bases and substitutions.

- **Finding genes and other functional elements.**
  These often change slower than other sequences.

Sequence alignment as an optimization problem

Goal of the sequence alignment: find pairs of homologous bases (coming from a common ancestor)

Modeling phase: choose a scoring scheme such that
– real alignments have high score
– false positives have low score

Optimization phase:
given two input sequences find the highest scoring alignment
– focus on computational efficiency

Problem formulation

Set up a **scoring scheme** for alignments

e.g. match +1, mismatch -1, gap -1

```
GAGAAGGCCATAATGACCTATGTGTCCAGCT
|||||| ||||    ||||| |||  || ||
GAGAAGTCCAT---CACCTACGTGGTCACCT
```

22 matches, 6 mismatches, 3 gaps → score 13.

In practice we often use more complex scoring schemes.

Problem 1: global alignment

Input: sequences $X = x_1x_2\dots x_n$ and $Y = y_1y_2\dots y_m$.

Output: alignment of X and Y with the highest score

Problem 2: local alignment

Input: sequences $X = x_1x_2\dots x_n$ and $Y = y_1y_2\dots y_m$.

Output: alignment of substrings $x_i\dots x_j$ and $y_k\dots y_\ell$ with highest score

Dynamic programming for global alignment (Needleman, Wunsch 1970)

Subproblem $A[i,j]$: highest score of a global alignment of $x_1x_2\dots x_i$ a $y_1y_2\dots y_j$

One of the strings has length 0: the other string is aligned to gaps
$A[0,j]=-j$, $A[i,0]=-i$

General case $i>0$, $j>0$:
if $x_i=y_j$ are aligned $A[i,j]=A[i-1,j-1]+1$
if $x_i\neq y_j$ are aligned $A[i,j]=A[i-1,j-1]-1$
if x_i is aligned to a gap $A[i,j]=A[i-1,j]-1$
if y_j is aligned to a gap $A[i,j]=A[i,j-1]-1$

$$\begin{array}{cc} x_1\dots x_{i-1} & x_i \\ \underbrace{y_1\dots y_{j-1}}_{A[i-1,j-1]} & \underbrace{y_j}_{\pm 1} \end{array} \qquad \begin{array}{cc} x_1\dots x_{i-1} & x_i \\ \underbrace{y_1\dots y_j}_{A[i-1,j]} & \underbrace{-}_{-1} \end{array} \qquad \begin{array}{cc} x_1\dots x_i & - \\ \underbrace{y_1\dots y_{j-1}}_{A[i,j-1]} & \underbrace{y_j}_{-1} \end{array}$$

Dynamic programming for global alignment

Subproblem $A[i,j]$: highest score of a global alignment of $x_1x_2\ldots x_i$ a $y_1y_2\ldots y_j$

General case $i > 0$**,** $j > 0$**:**

if $x_i = y_j$ are aligned $A[i,j] = A[i-1,j-1]+1$

if $x_i \neq y_j$ are aligned $A[i,j] = A[i-1,j-1]-1$

if x_i is aligned to a gap $A[i,j] = A[i-1,j]-1$

if y_j is aligned to a gap $A[i,j] = A[i,j-1]-1$

Recurrence:

$$A[i,j] = \max \begin{cases} A[i-1,j-1]+s(x_i,y_j), \\ A[i-1,j]-1, \\ A[i,j-1]-1 \end{cases}$$

where $s(x,y) = 1$ if $x = y$ and $s(x,y) = -1$ if $x \neq y$

Global alignment example

CATGTCGTA vs CAGTCCTAGA

		C	A	G	T	C	C	T	A	G	A
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
C	-1	1	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-2	0	2	1	0	-1	-2	-3	-4	-5	-6
T	-3	-1	1	1	?						
G	-4										
T	-5										
C	-6										
G	-7										
T	-8										
A	-9										

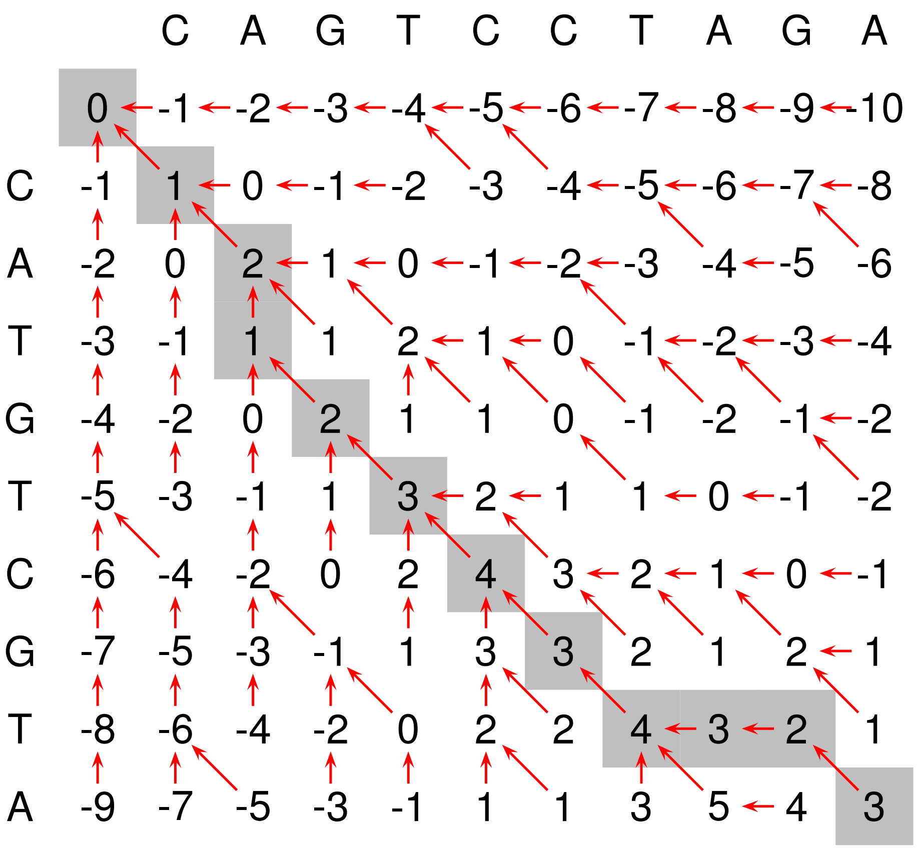
$$A[i,j] = \max \begin{cases} A[i-1,j-1] + s(x_i, y_j), \\ A[i-1,j] - 1, \\ A[i,j-1] - 1 \end{cases}$$

Global alignment example

CATGTCGTA vs CAGTCCTAGA

		C	A	G	T	C	C	T	A	G	A
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
C	-1	1	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-2	0	2	1	0	-1	-2	-3	-4	-5	-6
T	-3	-1	1	1	2	1	0	-1	-2	-3	-4
G	-4	-2	0	2	1	1	0	-1	-2	-1	-2
T	-5	-3	-1	1	3	2	1	1	0	-1	-2
C	-6	-4	-2	0	2	4	3	2	1	0	-1
G	-7	-5	-3	-1	1	3	3	2	1	2	1
T	-8	-6	-4	-2	0	2	2	4	3	2	1
A	-9	-7	-5	-3	-1	1	1	3	5	4	3

How to get the alignment?



		C	A	G	T	C	C	T	A	G	A
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
C	-1	1	0	-1	-2	-3	-4	-5	-6	-7	-8
A	-2	0	2	1	0	-1	-2	-3	-4	-5	-6
T	-3	-1	1	1	2	1	0	-1	-2	-3	-4
G	-4	-2	0	2	1	1	0	-1	-2	-1	-2
T	-5	-3	-1	1	3	2	1	1	0	-1	-2
C	-6	-4	-2	0	2	4	3	2	1	0	-1
G	-7	-5	-3	-1	1	3	3	2	1	2	1
T	-8	-6	-4	-2	0	2	2	4	3	2	1
A	-9	-7	-5	-3	-1	1	1	3	5	4	3

```
CA-GTCCTAGA
CATGTCGT--A
```

Dynamic programming for local alignment
(Smith, Waterman 1981)

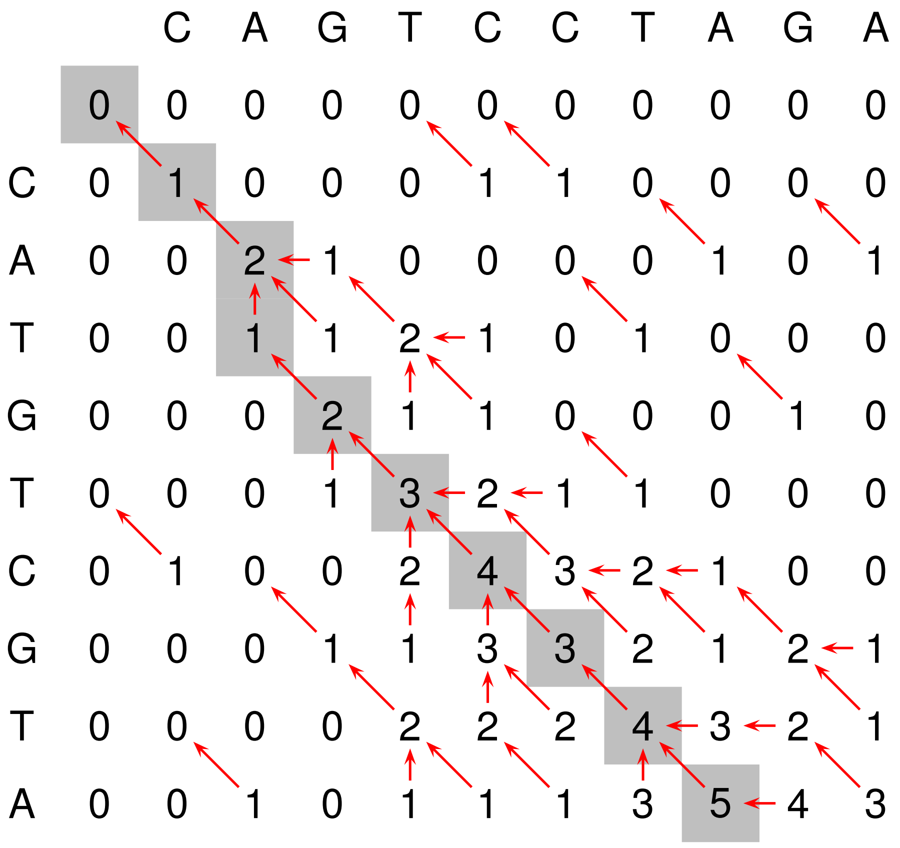
Subproblem $A[i,j]$: highest score of a local alignment of $x_1 x_2 \ldots x_i$ a $y_1 y_2 \ldots y_j$ that contains both x_i and y_j or is empty

One of the strings has length 0: $A[0,j] = A[i,0] = 0$ (empty aln.)

General case $i > 0$, $j > 0$:
if x_i and y_j are aligned $A[i,j] = A[i-1,j-1] + s(x_i, y_j)$
if x_i is aligned to a gap $A[i,j] = A[i-1,j] - 1$
if y_j is aligned to a gap $A[i,j] = A[i,j-1] - 1$
if x_i and y_j are not part of alignment with a positive score $A[i,j] = 0$

Recurrence: $$A[i,j] = \max \begin{cases} 0, \\ A[i-1,j-1] + s(x_i,y_j), \\ A[i-1,j] - 1, \\ A[i,j-1] - 1 \end{cases}$$

Example of local alignment



		C	A	G	T	C	C	T	A	G	A
	0	0	0	0	0	0	0	0	0	0	0
C	0	1	0	0	0	1	1	0	0	0	0
A	0	0	2	1	0	0	0	0	1	0	1
T	0	0	1	1	2	1	0	1	0	0	0
G	0	0	0	2	1	1	0	0	0	1	0
T	0	0	0	1	3	2	1	1	0	0	0
C	0	1	0	0	2	4	3	2	1	0	0
G	0	0	0	1	1	3	3	2	1	2	1
T	0	0	0	0	2	2	2	4	3	2	1
A	0	0	1	0	1	1	1	3	5	4	3

```
CA-GTCCTA
CATGTCGTA
```

More complex scoring schemes

Problems of the $+1, -1$ scoring scheme:

- Is really one mismatch or gap that bad compared to a single match?
- How to score protein alignments?
(20 element alphabet $\approx$ 200 parameters)

Goal of the scoring scheme:

- We want to distinguish better alignments from worse:
– Which arrangements of gaps are more meaningful?
- We want to know if an alignment has a biological meaning:
– Are the two sequences homologs or unrelated?

Probabilistic scoring scheme (the first attempt)

Assume X and Y are **correctly aligned homologs**

$a =$ probability that two bases form a **match**

$b =$ probability that two bases form a **mismatch**

$c =$ probability that a base is aligned to a **gap**

$a + b + c = 1$

Probability of alignment A:

```
GAGAAGGCCATAATGACCTATGTGTCCAGCT
|||||| ||||    ||||| |||  || ||
GAGAAGTCCAT---CACCTACGTGGTCACCT
```

$$\Pr(A) = a^{22}b^{6}c^{3}$$

Which alignment is more likely?

```
CACA
|  |
CCAA
```

$$\Pr(A) = a^2b^2$$

```
CACA-
| ||
C-CAA
```

$$\Pr(A) = a^3c^2$$

Probabilistic scoring scheme (the first attempt)

Take logarithm to change multiplication into addition
we can use S.-W. or N.-W. dynamic programming algorithms

$\Pr(A) = a^{22}b^6c^3$
$\log \Pr(A) = 22\log a + 6\log b + 3\log c$

Score: Match: $\log a$ Mismatch: $\log b$ Gap: $\log c$

Disadvantage of this scheme:

- Score always negative ⇒ how to do local alignment?
- Hard to compare different pairs of sequences

Scoring scheme based on two probabilistic models

Compare models H and R: “log likelihood ratio”

$$\log \frac{\Pr(X, Y \mid H)}{\Pr(X, Y \mid R)}$$

- Two sequences are **homologs**
  ⇒ likelihood ratio much higher than 1
  ⇒ positive score

- Two **unrelated** sequences
  ⇒ likelihood ratio much lower than 1
  ⇒ negative score

Scoring scheme based on two probabilistic models

(Ignore gaps for now)

Model H: Sequences X and Y are **correctly aligned homologs**

$\Pr(X,Y \mid H) = \prod_{i=1}^{n} p(x_i, y_i)$

$p(x_i, y_i)$: probability that alignment contains aligned bases x_i and y_i

Model R: Sequences X and Y are unrelated

$\Pr(X,Y \mid R) = \left(\prod_{i=1}^{n} p(x_i)\right) \left(\prod_{i=1}^{n} p(y_i)\right)$

$p(x_i)$: probability of occurrence of x_i in a sequence

Compare models H and R: “log likelihood ratio”

$$\log \frac{\Pr(X,Y \mid H)}{\Pr(X,Y \mid R)}$$

Scoring scheme based on two probabilistic models

$\Pr(X, Y \mid H) = \prod_{i=1}^{n} p(x_i, y_i)$

$\Pr(X, Y \mid R) = \left(\prod_{i=1}^{n} p(x_i)\right) \left(\prod_{i=1}^{n} p(y_i)\right)$

$$\log \frac{\Pr(X, Y \mid H)}{\Pr(X, Y \mid R)} = \log \frac{\prod_{i=1}^{n} p(x_i, y_i)}{\left(\prod_{i=1}^{n} p(x_i)\right) \left(\prod_{i=1}^{n} p(y_i)\right)} = \sum_{i=1}^{n} \log \frac{p(x_i, y_i)}{p(x_i)p(y_i)}$$

score for aligning bases x and y:

$$s(x, y) = \log \frac{p(x, y)}{p(x)p(y)}$$

BLOSUM62 protein scoring matrix

BLOcks of aminoacid **SU**bstitution **M**atrix; Henikoff, Henikoff 1992

- Choose **biologically relevant protein alignments** (BLOCKS)
- Only pairs with identity at most 62%
- $p(x,y)$: how often we see amino acids x and y aligned
- $p(x)$: how often we see amino acid x

- **Score for a pair of amino acids x and y**: $\log \dfrac{p(x,y)}{p(x)p(y)}$
- multiply by a constant and round to integers:
  - to avoid too big rounding error
  - integers allow faster computation

More complex scoring: Affine gap scores

```
CCCGACGAGAAGGCCATAATGACCTATGTGTCCAGCTTCTACCATGCCTTT
|| ||||||||| ||||    ||||| |||  || || ||| ||   ||||
CCGGACGAGAAGTCCAT---CACCTACGTGGTCACCTACTATCACTACTTT
```

Several consecutive gaps likely originated in a single mutation rather than each independently.

Penalty for starting a gap (gap opening cost) o,
Penalty for each next gap symbol (gap extension cost) e.
Gap of length g has penalty $o + e(g-1)$.
We choose $o < e$ (i.e. $|o| > |e|$).

Default settings of blastn: match +2, mismatch -3, $o = -5$, $e = -2$.
Example above: 22 matches, 6 mismatches, 1 gap of length 3
→ score $2 \cdot 22 - 3 \cdot 6 - 5 - 2 \cdot 2 = 16$.

Summary

- Global and local alignments
- Needleman-Wunsch and Smith-Waterman algorithms
- Scoring schemes for alignments based on comparing likelihoods
- Protein BLOSUM scoring matrix
- Affine gap penalties

Problems to think about:

1. **Running time of Smith-Waterman:** $O(nm)$

   n - length of the first sequence

   m - length of the second sequence

   Local alignments between human and mouse?

2. We found an alignment with score 14

   Is this a good score or is it a score that would appear just by chance?

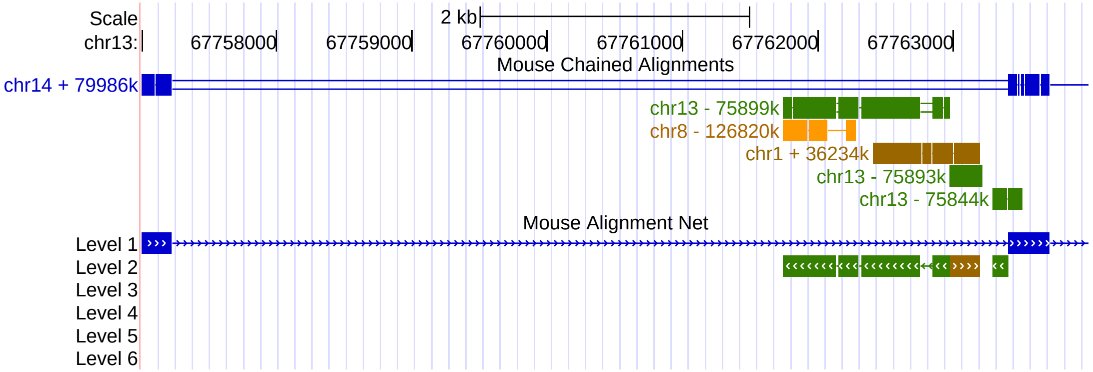
Announcements

- Submit your preferences for journal club papers using the form at the website until next Wednesday, Oct. 20 22:00
- Homework 1 will be published on the website, submit until Tuesday November 9 22:00 (pdf via Moodle, guests by e-mail to brejova@dcs.fmph.uniba.sk)
- You are are allowed to discuss homework questions with classmates, but do not take notes during discussions and do not show your solutions to others. Everybody should write their homework submission independently, do not copy from classmates or other sources.
- Please use MS Teams for questions regarding homeworks, quizzes and the course in general.
- However, any questions involving your ideas about solving the questions should be sent privately to instructors by email.

Sequence alignment 2/2

Tomáš Vinař

October 14 2021



Summary from the last lecture

- **Global and local alignment problem**

  Input: sequences $X = x_1x_2 \dots x_n$ and $Y = y_1y_2 \dots y_m$.

  Output: alignment of X and Y with the highest score

  or alignment of **substrings** $x_i \dots x_j$ a $y_k \dots y_\ell$ with the highest score

- **Correct algorithms** using dynamic programming

- **Realistic scoring schemes**

We have dynamic programming, what else do we need?

Running time: $O(n^2)$ on two sequences of length n

How much is that in practice?

(simple implementation, random sequences, desktop computer)

n	time
100	0.0008s
1,000	0.08s
10,000	8s
100,000	13m (*)
1,000,000	22h (*)
10,000,000	3months (*)
100,000,000	25years (*)

We need a more efficient algorithm, particularly for comparative genomics

Memory: basic implementation $O(n^2)$, but can be done in $O(n)$

Heuristic alignment

- Trade sensitivity for speed (some alignments not found)
- Reduce the search to “promising” parts of the matrix

Heuristic local alignment

BLASTN [Altschul et al 1990], FASTA [Pearson 1988]

- Find short exact matches of length w (**seeds**)
- Extend hits along diagonals to ungapped alignments
- Connect alignments on nearby diagonals to gapped alignment
- Possibly optimize by dynamic programming

How to find short exact matches?

- Create a **dictionary** of short substrings of length w from the first sequence.
- Search for all substring from the second sequence in the dictionary

Exmple: CAGTCCTAGA vs CATGTCATA

Dictionary:

AG	2, 8
CA	1
CC	5
CT	6
GA	9
GT	3
TA	7
TC	4

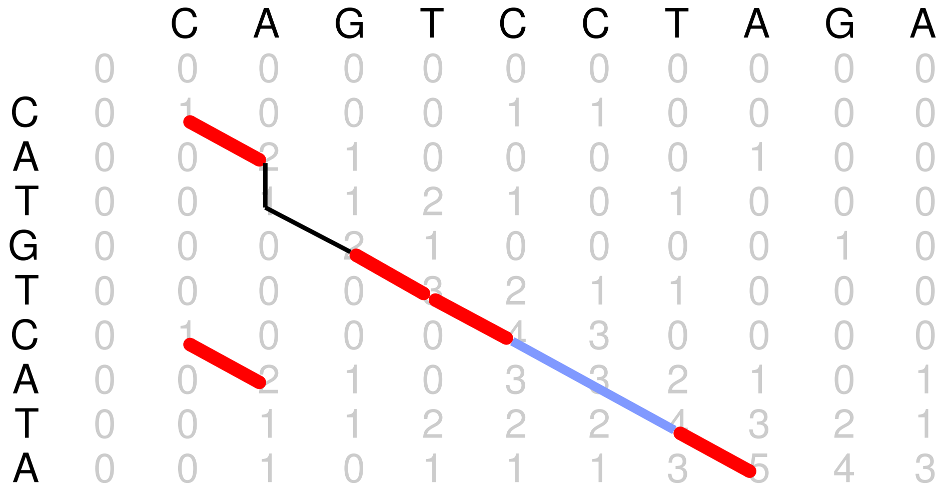
Search for:

CA	→	1
AT	→	-
TG	→	-
GT	→	3
TC	→	4
CA	→	1
AT	→	-
TA	→	7

Heuristic local alignment

Example: start from **seeds** of length $w = 2$

(in practice we would use $w = 11$ or more)



		C	A	G	T	C	C	T	A	G	A
	0	0	0	0	0	0	0	0	0	0	0
C	0	1	0	0	0	1	1	0	0	0	0
A	0	0	2	1	0	0	0	0	1	0	0
T	0	0	1	1	2	1	0	1	0	0	0
G	0	0	0	2	1	0	0	0	0	1	0
T	0	0	0	0	3	2	1	1	0	0	0
C	0	1	0	0	0	4	3	0	0	0	0
A	0	0	2	1	0	3	3	2	1	0	1
T	0	0	1	1	2	2	2	4	3	2	1
A	0	0	1	0	1	1	1	3	5	4	3

1. find hits
2. ungapped
3. gapped

Running time of heuristic local alignment

Algorithm

- Find seeds (short exact matches of length w)
- **Expensive step:** extend/connect seeds to longer alignments

Random seeds of length w: not part of any high-scoring alignment. These are filtered in the extension step, but they slow down the program

How many random hits?

Two unrelated nucleotides match with probability $1/4$
We have w matches in a row with probability 4^{-w}
Expected number of false positives roughly $nm4^{-w}$
Increase of w by 1 means cca 4-fold decrease of spurious seeds

Sensitivity of heuristic local alignment

Algorithm

- Find seeds (short exact matches of length w)
- **Expensive step:** extend/connect seeds to longer alignments

Some alignments not found: high score but no seed of length w

Example:

```
CA-GTCCTA
CATGTCATA
```

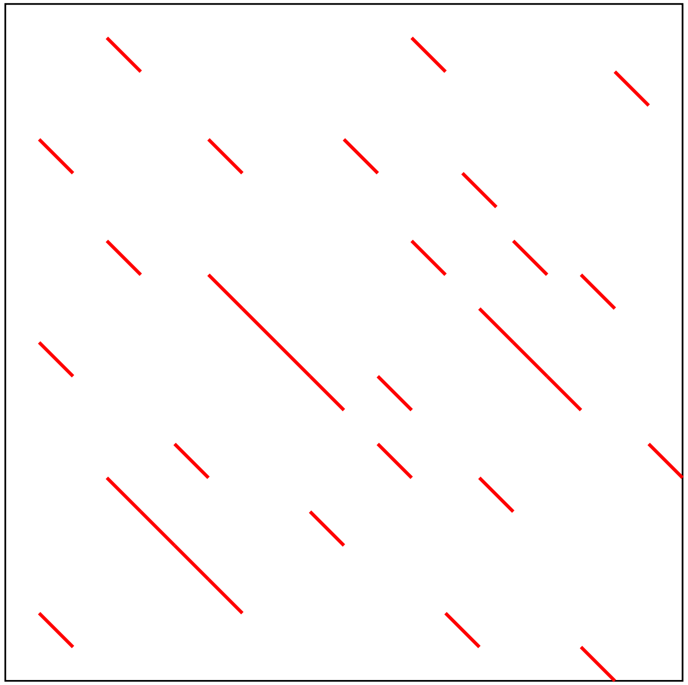
no seed of length $w \geq 4$

Sensitivity: fraction of real alignments containing a seed of length w

Sensitivity vs. running time

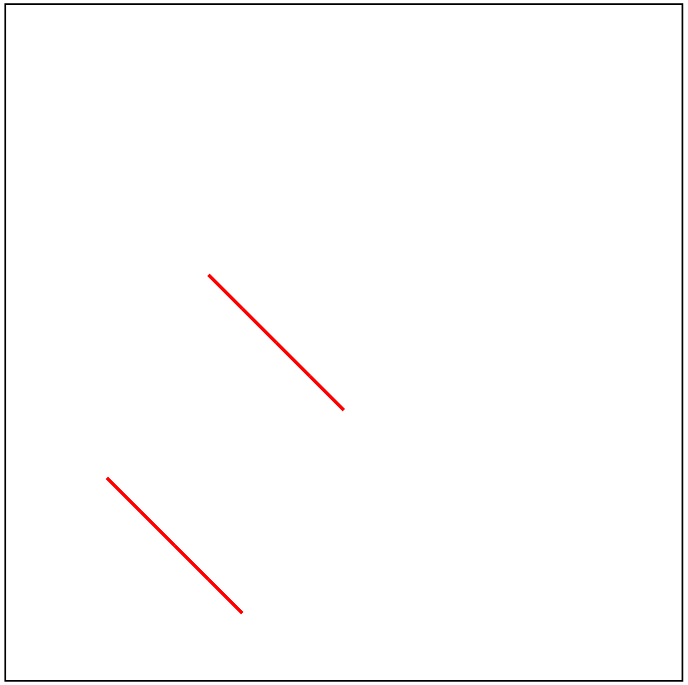
Small w

many spurious seeds, slow



Large w

many alignments not found

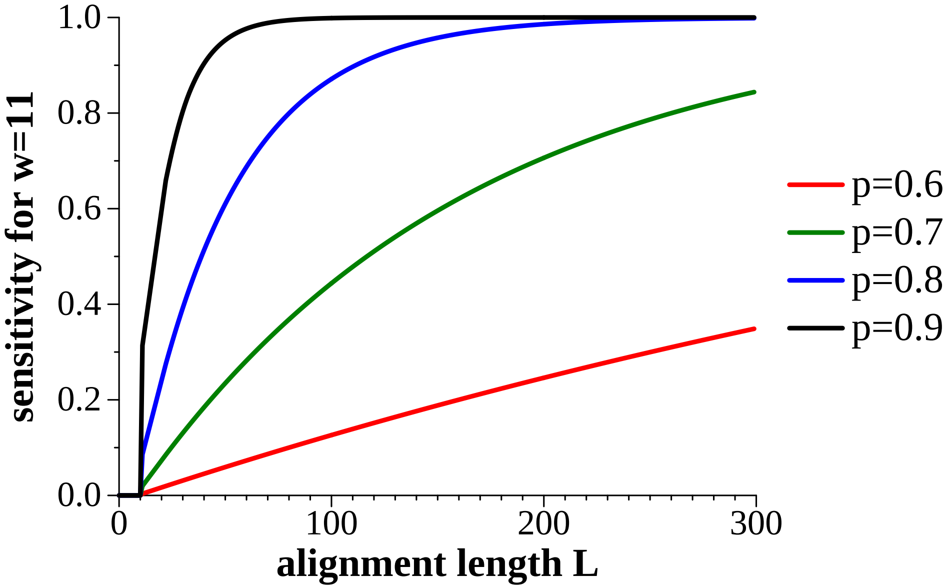


Can we estimate the sensitivity?

Assume random ungapped alignment of length L

Every position match with probability p

Sensitivity $f(L,p) = \Pr($alignment contains w consecutive matches$)$



(human-mouse: $p \approx 0.7$)

Protein BLAST

BLOSUM62 scoring matrix for proteins

Instead of exact match of length w, protein BLAST requires 3 amino acids with score at least 13

Hit:		Not a hit:	

	`N I R`
	`N L R`
			`A I L`
			`A I L`

	6+2+5=13		4+4+4=12

Examples of software tools for various tasks

NCBI BLAST: blastn for DNA/RNA, blastp for proteins, tblastx translates DNA to proteins and uses blastp

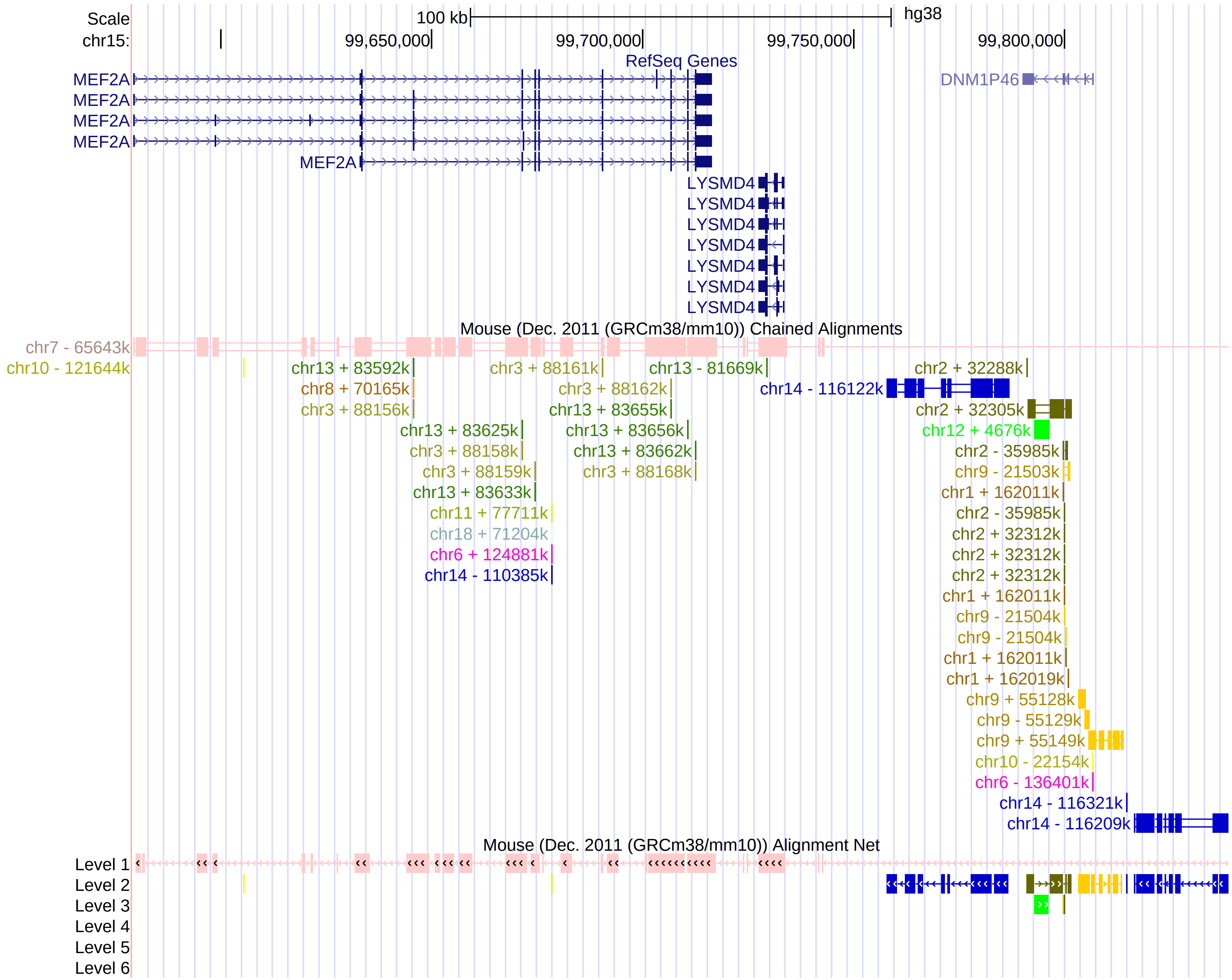
UCSC Blat: very fast search for very similar sequences, i.e. aligning sequencing reads to the genome

- uses very large values of w
- can split alignments with big gaps (aligning transcripts with introns)

Whole-genome alignments

For each section of human genome find closest section from mouse, dog, chicken, etc. (see e.g. UCSC genome browser)

- Local alignments will cover protein coding exons and other conserved parts
- Sections that diverged too much cannot be aligned
- If there was a duplication, we need to decide which pairs belong together
- **Synteny principle:** if two similar sections (local alignments) are present in the same order and orientation in two genomes, they likely evolved from the same common ancestor (orthologs)

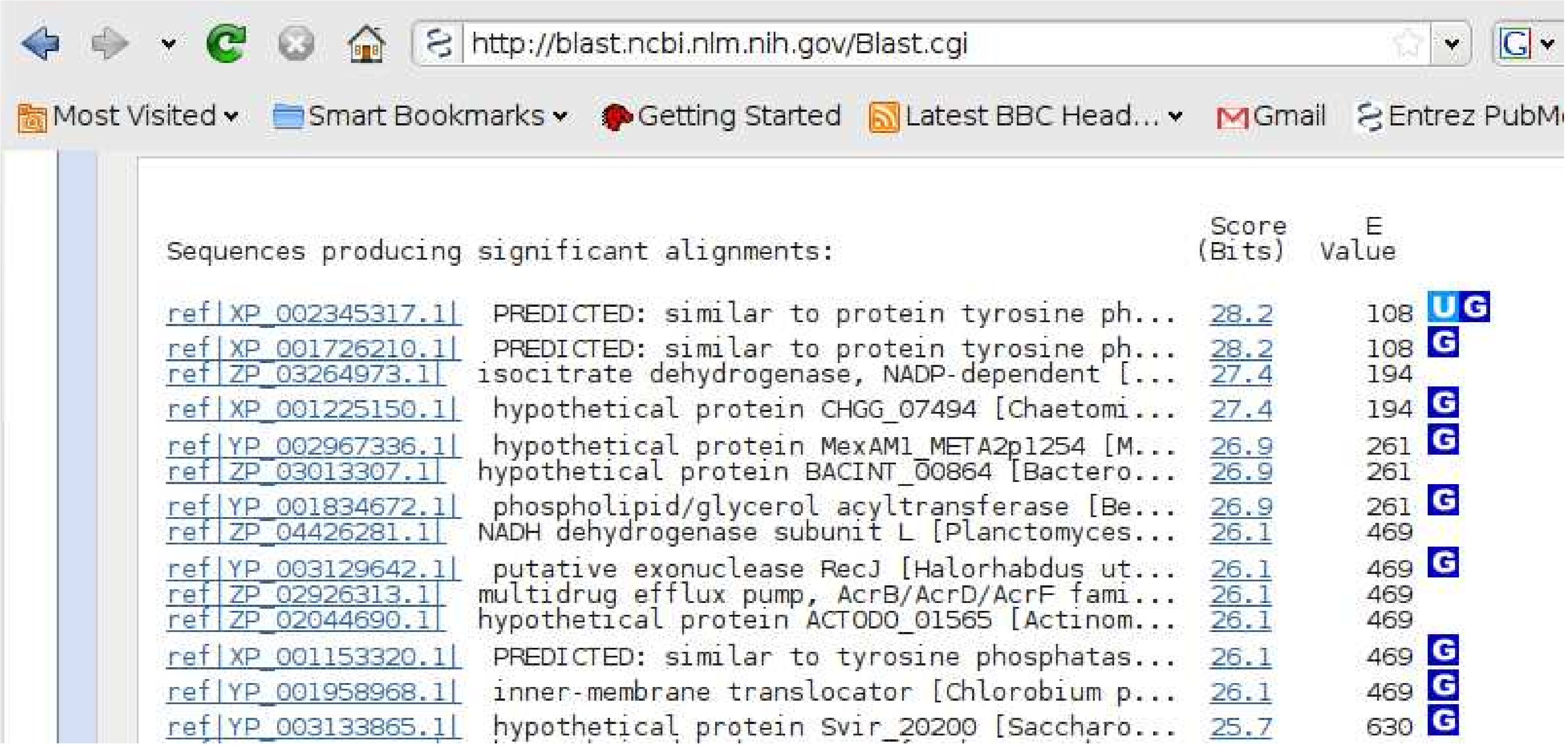

Scale
100 kb
hg38
chr15:
99,650,000
99,700,000
99,750,000
99,800,000
RefSeq Genes
MEF2A
MEF2A
MEF2A
MEF2A
MEF2A
DNM1P46
LYSMD4
LYSMD4
LYSMD4
LYSMD4
LYSMD4
LYSMD4
LYSMD4
Mouse (Dec. 2011 (GRCm38/mm10)) Chained Alignments
chr7 - 65643k
chr10 - 121644k
chr13 + 83592k
chr8 + 70165k
chr3 + 88156k
chr13 + 83625k
chr3 + 88158k
chr3 + 88159k
chr13 + 83633k
chr11 + 77711k
chr18 + 71204k
chr6 + 124881k
chr14 - 110385k
chr3 + 88161k
chr3 + 88162k
chr13 + 83655k
chr13 + 83656k
chr13 + 83662k
chr3 + 88168k
chr13 - 81669k
chr14 - 116122k
chr2 + 32288k
chr2 + 32305k
chr12 + 4676k
chr2 - 35985k
chr9 - 21503k
chr1 + 162011k
chr2 - 35985k
chr2 + 32312k
chr2 + 32312k
chr2 + 32312k
chr1 + 162011k
chr9 - 21504k
chr9 - 21504k
chr1 + 162011k
chr1 + 162019k
chr9 + 55128k
chr9 - 55129k
chr9 + 55149k
chr10 - 22154k
chr6 - 136401k
chr14 - 116321k
chr14 - 116209k
Mouse (Dec. 2011 (GRCm38/mm10)) Alignment Net
Level 1
Level 2
Level 3
Level 4
Level 5
Level 6

Multiple sequence alignment

Running time: $O(2^k n^k)$ for k sequences of length n

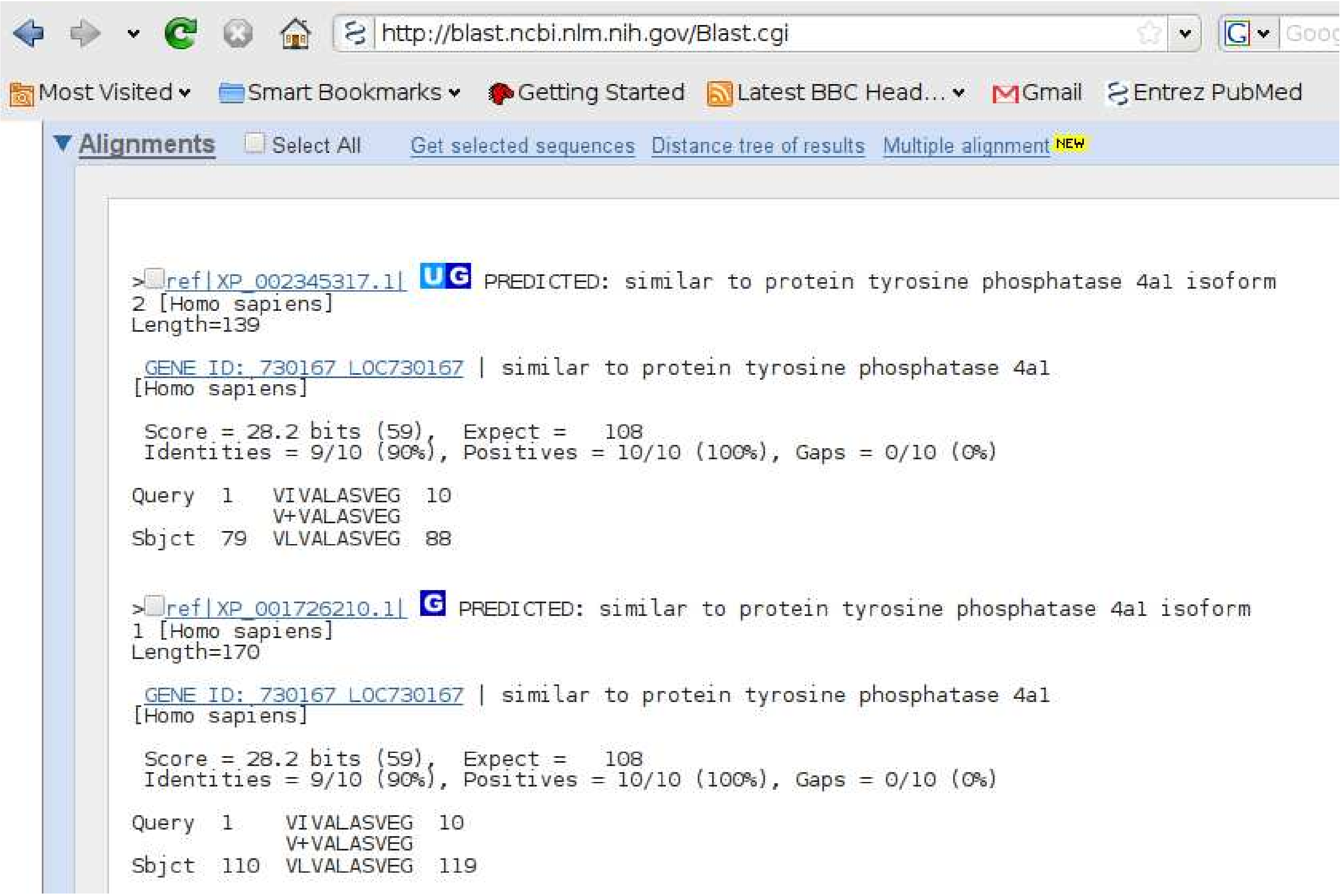
For general k NP-hard.

Heuristic algorithms, e.g. CLUSTAL-W [Higgins et al., 1996], MUSCLE [Edgar, 2004] and TBA [Blanchette et al., 2004].



```
                                                                 Score     E
Sequences producing significant alignments:                     (Bits)  Value

ref|XP_002345317.1|  PREDICTED: similar to protein tyrosine ph...  28.2     108  U G
ref|XP_001726210.1|  PREDICTED: similar to protein tyrosine ph...  28.2     108  G
ref|ZP_03264973.1|  isocitrate dehydrogenase, NADP-dependent [...  27.4     194
ref|XP_001225150.1|  hypothetical protein CHGG_07494 [Chaetomi...  27.4     194  G
ref|YP_002967336.1|  hypothetical protein MexAM1_META2p1254 [M...  26.9     261  G
ref|ZP_03013307.1|  hypothetical protein BACINT_00864 [Bactero...  26.9     261
ref|YP_001834672.1|  phospholipid/glycerol acyltransferase [Be...  26.9     261  G
ref|ZP_04426281.1|  NADH dehydrogenase subunit L [Planctomyces...  26.1     469
ref|YP_003129642.1|  putative exonuclease RecJ [Halorhabdus ut...  26.1     469  G
ref|ZP_02926313.1|  multidrug efflux pump, AcrB/AcrD/AcrF fami...  26.1     469
ref|ZP_02044690.1|  hypothetical protein ACTODO_01565 [Actinom...  26.1     469
ref|XP_001153320.1|  PREDICTED: similar to tyrosine phosphatas...  26.1     469  G
ref|YP_001958968.1|  inner-membrane translocator [Chlorobium p...  26.1     469  G
ref|YP_003133865.1|  hypothetical protein Svir_20200 [Saccharo...  25.7     630  G
```



```
>□ref|XP_002345317.1| UG PREDICTED: similar to protein tyrosine phosphatase 4a1 isoform
2 [Homo sapiens]
Length=139

 GENE ID: 730167 LOC730167 | similar to protein tyrosine phosphatase 4a1
[Homo sapiens]

 Score = 28.2 bits (59),  Expect =   108
 Identities = 9/10 (90%), Positives = 10/10 (100%), Gaps = 0/10 (0%)

Query  1   VIVALASVEG  10
           V+VALASVEG
Sbjct  79  VLVALASVEG  88


>□ref|XP_001726210.1| G PREDICTED: similar to protein tyrosine phosphatase 4a1 isoform
1 [Homo sapiens]
Length=170

 GENE ID: 730167 LOC730167 | similar to protein tyrosine phosphatase 4a1
[Homo sapiens]

 Score = 28.2 bits (59),  Expect =   108
 Identities = 9/10 (90%), Positives = 10/10 (100%), Gaps = 0/10 (0%)

Query  1    VIVALASVEG  10
            V+VALASVEG
Sbjct  110  VLVALASVEG  119
```

How to distinguish when the alignment is “real”?

Query length m. Database length n.
Alignment with score S.

P-value: Probability that a random query of length m in a random database of length n yields alignment of score at least S

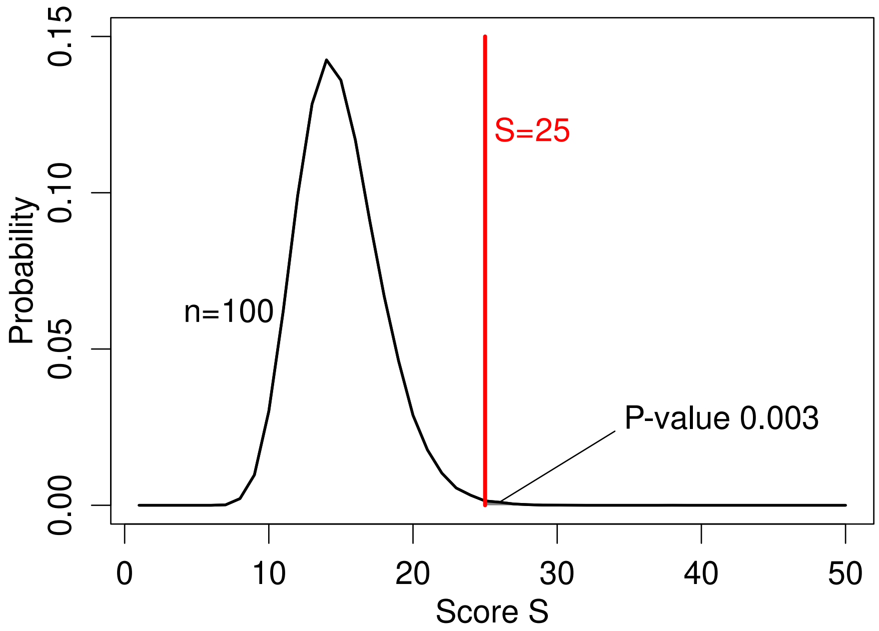
E-value: Expected number of alignments with the score of at least S when searching for a random query of length m in a random database of length n

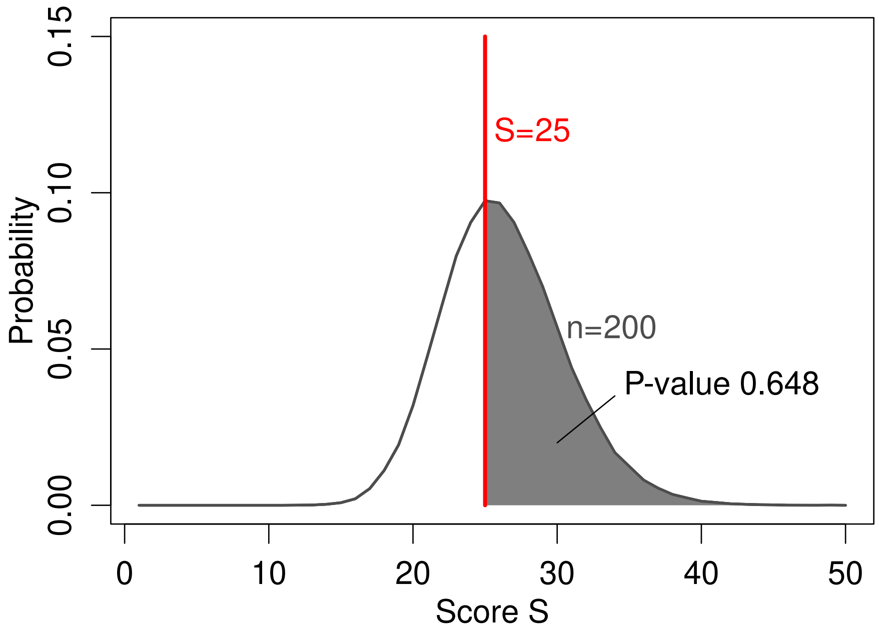
Note: $P = 1 - e^{-E} \Rightarrow$ for very small values of E, $P \approx E$

[Karlin and Altschul, 1990, Dembo et al., 1994]

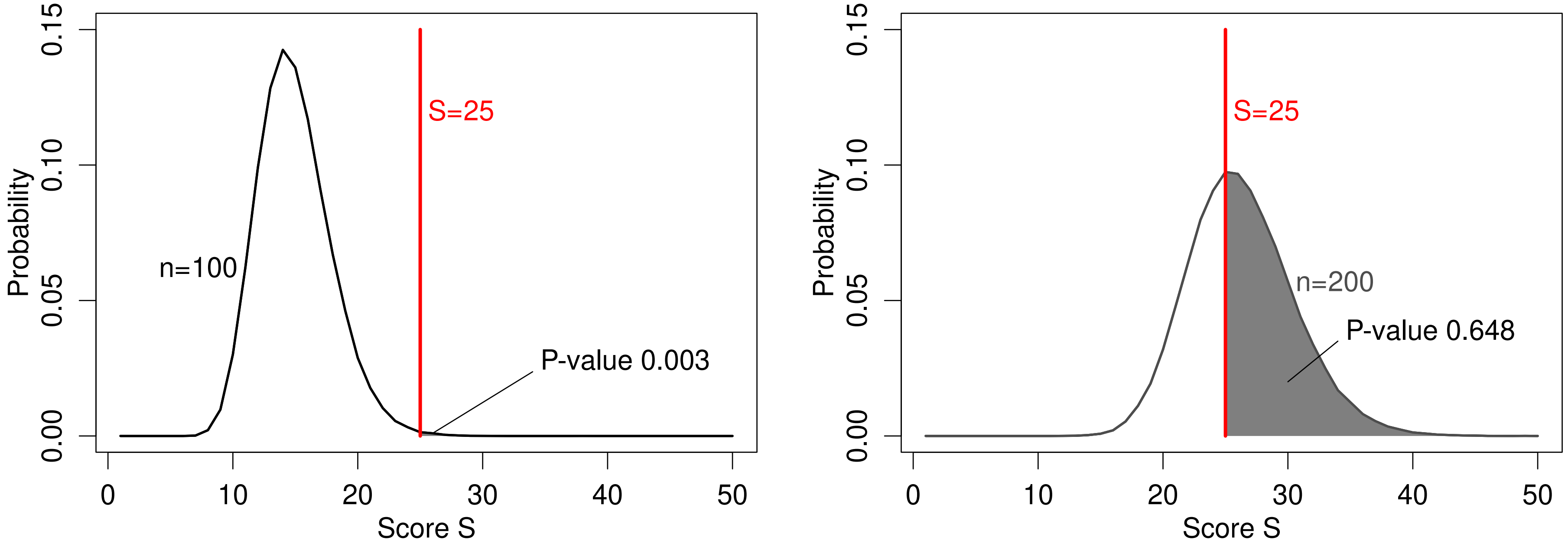
Computing P-values by simulation

- Generate a random query and a random database of length n
- Compute best local alignment (+1/-1 scheme)
- Record the resulting score
- Repeat many times





Computing P-values by simulation (cont)



P-value for score 25:

How many alignments have score 25 or higher?

(In practice, simulations are slow, but we have mathematical estimates of how these distributions look like.)

Summary

- Sequence alignment is the essential bioinformatics tool
- Problem formulation: defining a scoring scheme
- Problem solution: either slow and exact algorithms, or fast heuristics that can miss some alignments
- There are specialized tools for various tasks related to the sequence alignment
- Estimation of statistical significance (P-values) is an important tool in distinguishing real alignments from those that occur just by chance

Announcements

- Homework 1 is published on the website, submit until Tuesday November 9 22:00

Journal club: groups

- Groups published on the course website
- MS teams has a channel for each group, use it to communicate within group (chat, online meetings, document sharing)
- Group 4 has three members who do not speak Slovak, two of them are not located in Bratislava

Journal club: meeting

- Everybody first reads the assigned paper individually,
  then organize a group meeting, where you discuss the paper
  (particularly any portions which you did not understand),
  plan writing of the journal club report
- The first group meeting should occur no later than Nov. 23.
  It can take place in MS Teams or in person.
- Announce the first meeting at 1 day in advance
  (time and location or link) in the group channel chat
- After the meeting, post a short summary to the group channel:
  who participated, what did you agree upon, any problems
- You can arrange a consultation with us if needed.
- You do not need to report any additional meetings.

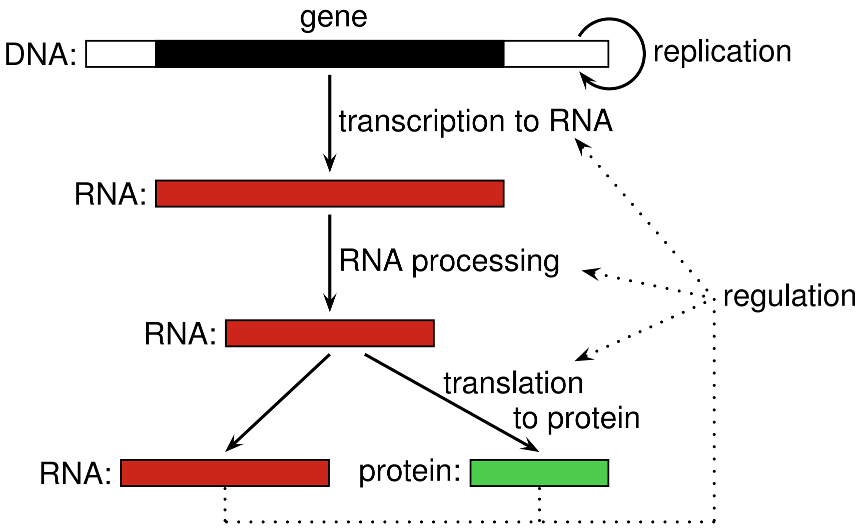
Journal club report

- The main methods and results of the article in your own words
- Understandable for students of this course (both computer scientists and biologists)
- You do not have to cover the entire content of the article in the report and, conversely, you can use other resources
- Try to express your own view of the topic, do not strictly follow the text of the article
- The recommended length is about 1-2 pages per person, one coherent text
- The report should list the members of the group who have actively participated. They will get the same points (the rest zero)

Gene finding

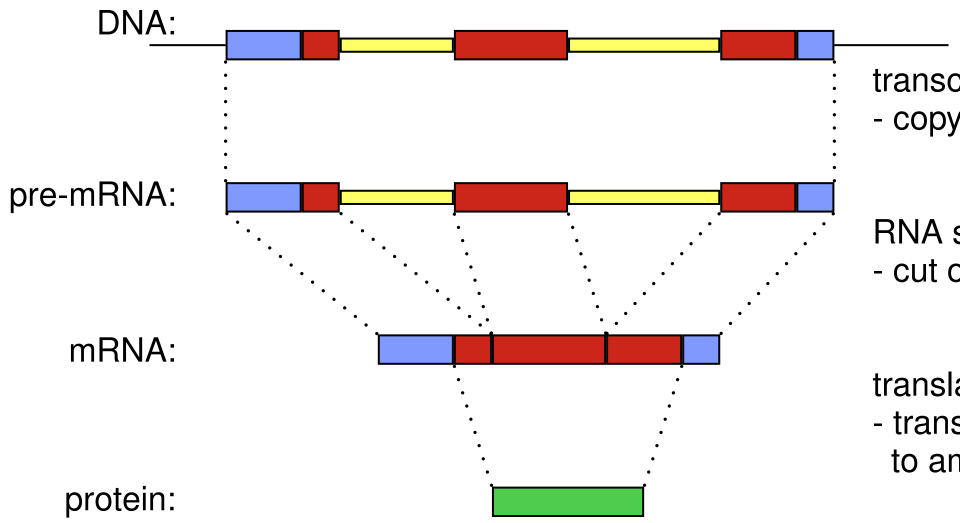
Tomáš Vinař
October 21, 2021

What to do with sequenced and assembled genomes?



- protein-coding genes (today's lecture)
- RNA genes
- signals for regulation of transcription, splicing, etc.
- pseudogenes (non-functional copies of genes)
- sequence repeats

Protein synthesis and translation in Eukaryotes



transcription
- copy contiguous region of DNA

RNA splicing
- cut out introns

translation
- translate codons, triples of nucleotides,
  to aminoacids

exon:

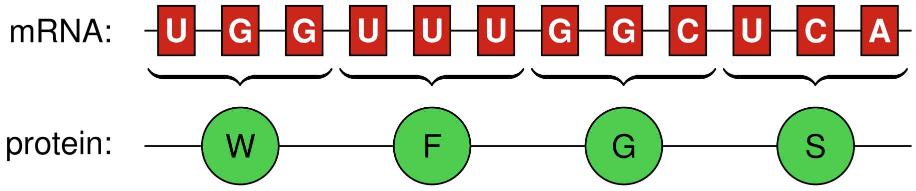
🟦 UTR (untranslated region)

🟥 coding exon

🟨 intron

— intergenic region

Translation: three nucleotides (codon) → aminoacid in the protein



Human genome

- protein-coding genes
  - cca 20,000, cover approx. 40% of genome length
  - cca 10 exons in each gene
  - exons cover approx. 2% of genome length
  - coding exons approx. 1.2%

- sequence repeats
  - cover approx. 49% of genome length

Bioinformatics problem: Gene finding

Goal: find all protein-coding genes in the genome
(assemble a catalogue of all proteins)

Simplifying assumpations:

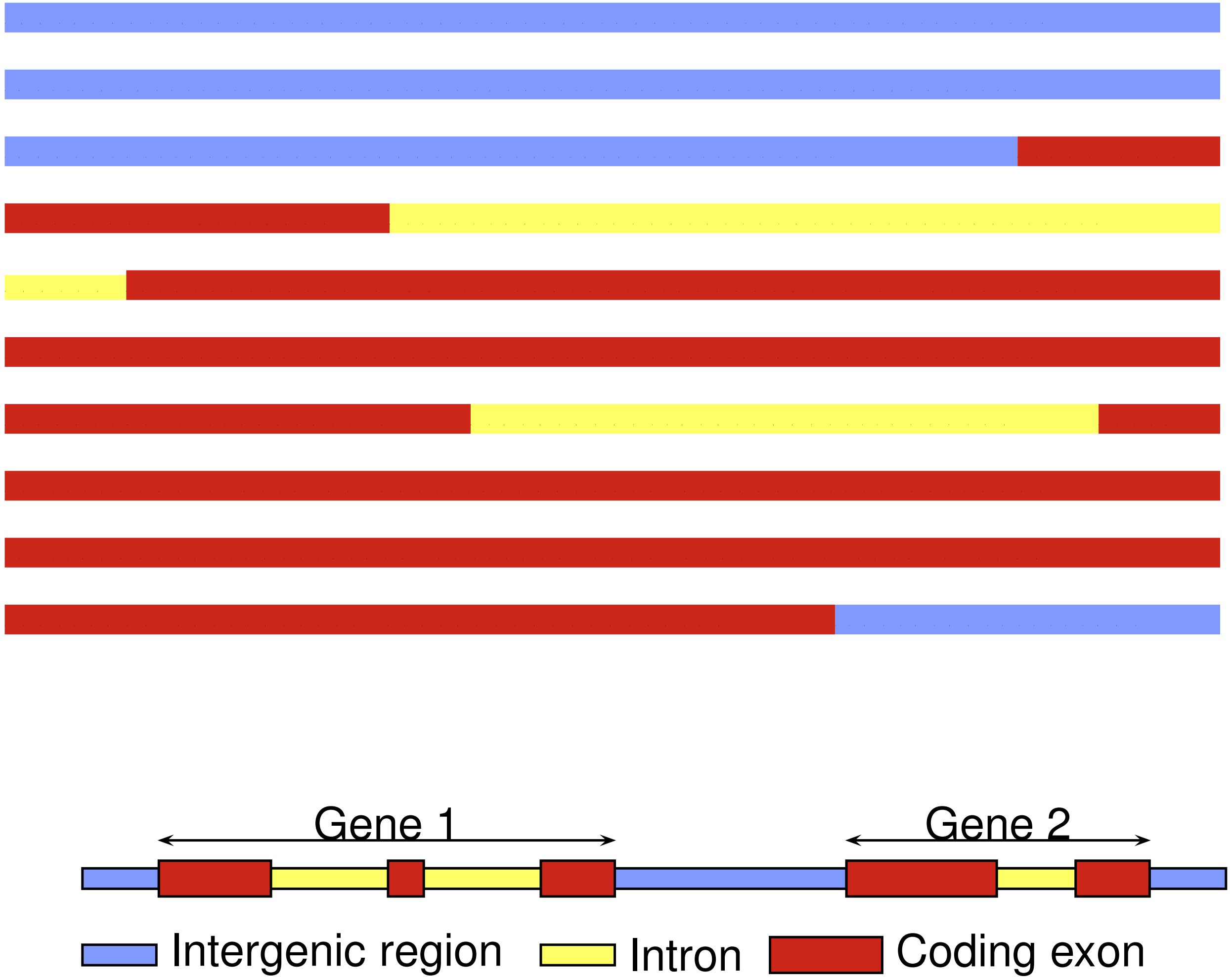
- no alternative splicing, no overlapping genes
- we are not searching for untranslated regions (UTRs) at the beginning and the end of the gene

Bioinformatics problem: Gene finding

Input: DNA sequence

Bioinformatics problem: Gene finding

Goal: mark each base as intron/exon/intergenic



Bioinformatics problem: gene finding

Input: DNA sequence

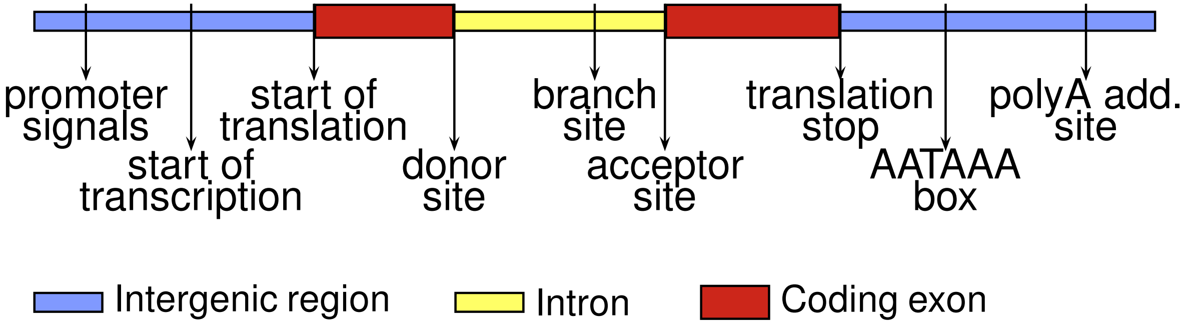
Goal: mark each base as intron/exon/intergenic

- Still not a well-defined problem!
  How to recognize a gene?

How to recognize a gene?

Signals at the exon boundaries:

short strings that serve as binding sites for the transcription machinery



Example of a signal: donor splice site

```
           Exon                   Intron
<----------------------><---------------------->
ccatcccctatatttatggcagGTgaggaaagggtgggggctgggg
attcatcatcatgggtgcatcgGTgagtatctcccaggccccaatc
agaagatctaccccaccatctgGTaagtgtgtcccaccactgcccc
acagagtgagcccttcttcaagGTgggtggtgtcagggcctccccc
acgagtcctgcatgagccagatGTaaggcttgccgttgccctccct
tgcagaacctcatggtgctgagGTggggccaagcctgggccggggg
tcgatgaatttgggatcatccgGTgagagctcttcctctctcctgg
agatgacgtccgtgatgagaagGTagggggtgcaccccagtcccca
gtggagaatgagaggtgggatgGTaggtgatgccttcgaggcccag
tttcttgtggctattttaaaagGTaattcatggagaaatagaaaaa
```

How to recognize a gene?

Sequence composition:

- different k-mer frequency in coding and non-coding regions,
- coding regions are 3-periodic,
- stop codons (TAA, TGA, TAG) appear only at the end of the last exon.

Example: in human genome, exons are GC rich

		a	c	g	t
coding exon	0	0.26	0.26	0.32	0.16
	1	0.30	0.24	0.20	0.26
	2	0.17	0.32	0.31	0.20
intron	🟨	0.26	0.22	0.22	0.30
intergenic	🟦	0.27	0.23	0.23	0.27

Bioinformatics problem: Gene finding

Input: DNA sequence

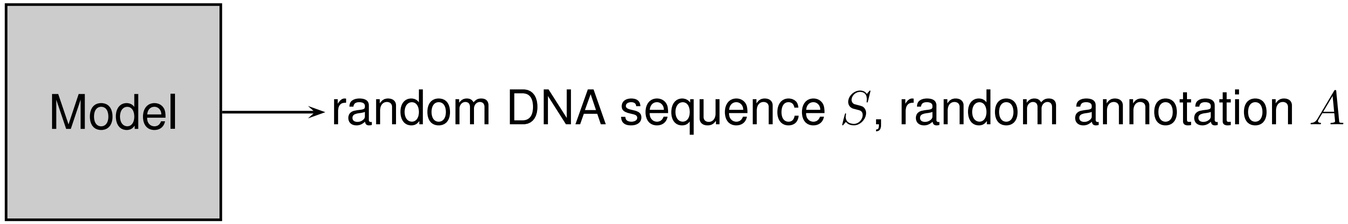
Goal: mark each base as intron/exon/intergenic

- Not a well-defined problem!
  How to recognize a gene?
- No information **by itself** can uniquely determine which parts of the sequence correspond to genes.
- Want a **scoring scheme** that will tell us how well a particular annotation corresponds to our knowledge about gene structure.
- Then we are looking for an annotation (or a segmentation of the sequence into non-overlapping regions representing individual genes) with the **maximum score.**
- We use **probabilistic models** to define such scoring scheme.

Probabilistic model of protein-coding genes

No single source of information uniquely determines genes

Combine all sources using probablistic models



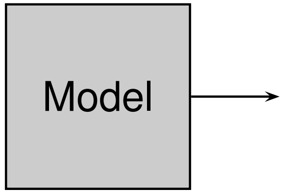
Model → random DNA sequence S, random annotation A

$\Pr(S,A)$ – probability model generates pair (S,A).

Model with high probability generates pairs with properties similar to the real genes

Using a probabilistic model: for a new sequence S find the most probable annotation $A = \arg\max_A \Pr(A|S)$

Probabilistic model of protein-coding genes



random DNA sequence S, random annotation A

Using a probabilistic model: for a new sequence S find the most probable annotation $A = \arg\max_A \Pr(A|S)$

Toy example: sequences of length 2

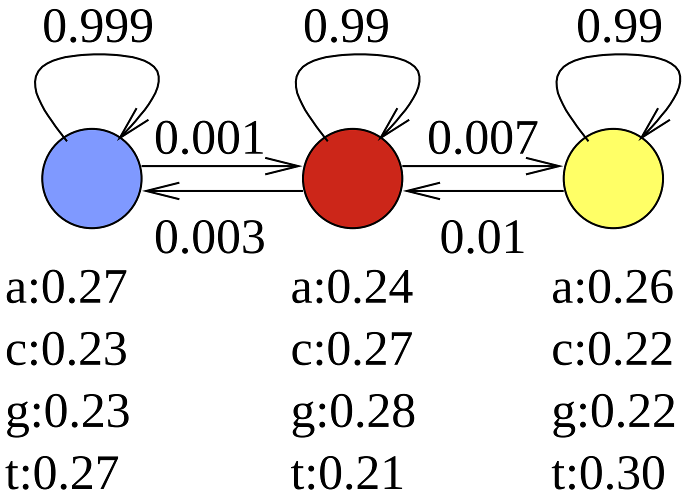
Table of probabilities for 16 sequences, 9 annotations (sums to 1)

The most probable annotation for $S =$aa is aa 🟦🟥.

aa 🟦🟦	0.008	ac 🟦🟦	0.009	ag 🟦🟦	0.0085	. . .
aa 🟦🟨	0	ac 🟦🟨	0	. . .		
aa 🟦🟥	0.011	. . .				
aa 🟨🟦	0					
aa 🟨🟨	0.009					
aa 🟨🟥	0					
aa 🟥🟦	0.007					
aa 🟥🟨	0					
aa 🟥🟥	0.010					

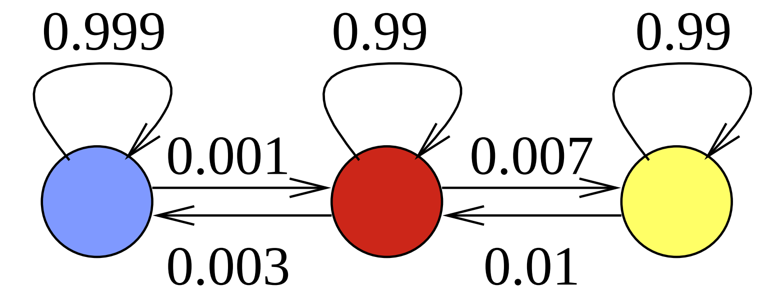
Hidden Markov model (HMM)

Way of defining models for longer sequences.



- Finite-state automaton, states e.g. exon, intron, intergenic
- Generates sequences and annotations base-by-base
- In each step, in the current state, randomly generate a single base according to the state's emmission table
- Then randomly transition to the next state according to the probabilities on edges (transition probabilities)

Hidden Markov model (HMM)



a:0.27	a:0.24	a:0.26
c:0.23	c:0.27	c:0.22
g:0.23	g:0.28	g:0.22
t:0.27	t:0.21	t:0.30

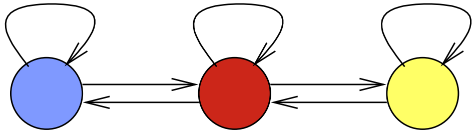
Assume the model starts in the blue state

Example:

Pr(🟦🟥🟥) $= 0.27 \cdot 0.001 \cdot 0.27 \cdot 0.99 \cdot 0.24 = 0.000017$

Pr(🟦🟦🟦) $= 0.27 \cdot 0.999 \cdot 0.23 \cdot 0.999 \cdot 0.27 = 0.017$

Notation



Sequence $S_1, \ldots, S_n$

Annotation $A_1, \ldots, A_n$

Model parameters:

Transition probability $a(u, v) = \Pr(A_{i+1} = v | A_i = u)$,

Emission probability $e(u, x) = \Pr(S_i = x | A_i = u)$,

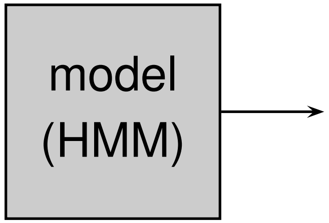
Starting probability $\pi(u) = \Pr(A_1 = u)$.

a	🟥	🟨	🟦
🟥	0.99	0.007	0.003
🟨	0.01	0.99	0
🟦	0.001	0	0.999

e	a	c	g	t
🟥	0.24	0.27	0.28	0.21
🟨	0.26	0.22	0.22	0.30
🟦	0.27	0.23	0.23	0.27

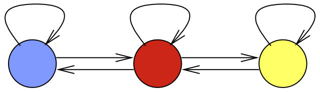
Resulting probability: $\Pr(A_1, \ldots, A_n, S_1, \ldots, S_n) =$
$\pi(A_1)e(A_1, S_1) \prod_{i=2}^{n} a(A_{i-1}, A_i)e(A_i, S_i)$

Finding genes with HMMs



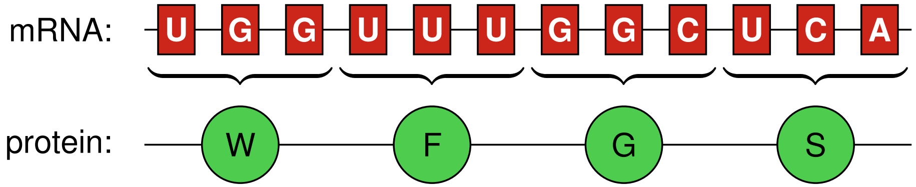
model (HMM) → random DNA sequence S, random annotation A (similar to real DNA)

$\Pr(S, A)$ – probability that the model generates pair (S, A).

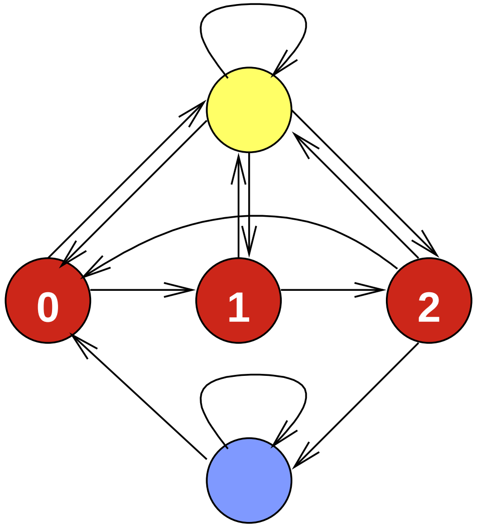
- **Determine states and transitions of the model:** by hand based on your knowledge about the gene structure 
- **Parameter training:** emission and transition probabilities are determined based on the real sequences with known genes (**training set**)
- **Use:** for a new sequence S, find the most probable annotation $A = \arg\max_A \Pr(A|S)$ Viterbi algorithm in $O(nm^2)$ (dynamic programming)

Gene finding HMM: 3-periodic exons

three nucleotides (codon) → aminoacid in the protein



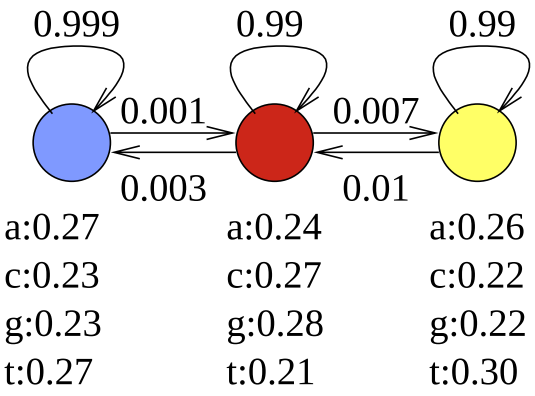
Instead of a single state for exon, use three states in a cycle

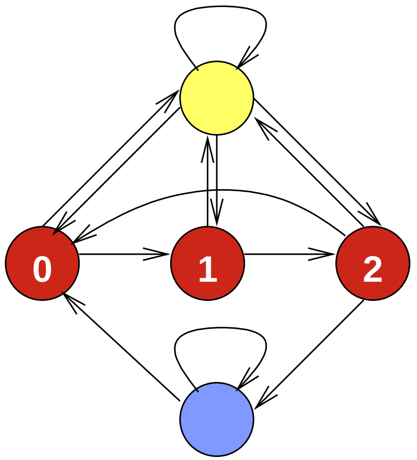


a	0	1	2	🟨	🟦
0	0		0		0
1	0	0			0
2		0	0		
🟨					0
🟦		0	0	0	

$\Pr(A_i | A_{i-1})$

Emission probabilities of new states will differ



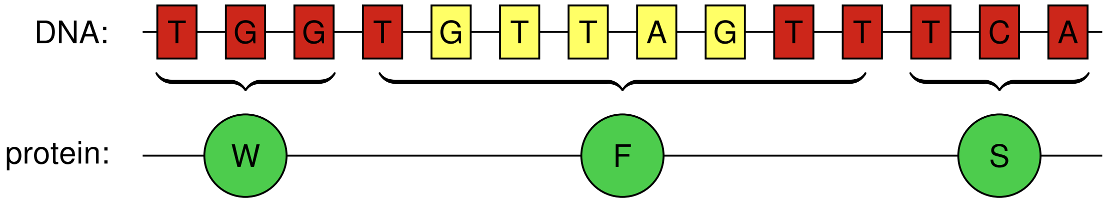


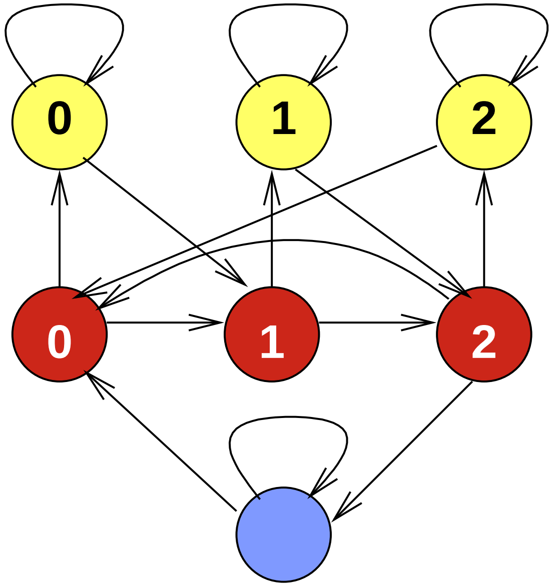
e	a	c	g	t
🟥	0.24	0.27	0.28	0.21
🟨	0.26	0.22	0.22	0.30
🟦	0.27	0.23	0.23	0.27

e	a	c	g	t
0	0.26	0.26	0.32	0.16
1	0.30	0.24	0.20	0.26
2	0.17	0.32	0.31	0.20
🟨	0.26	0.22	0.22	0.30
🟦	0.27	0.23	0.23	0.27

Gene finding HMM: consistent codons

Intron can interrupt a codon in the middle, but we must continue where we left off.

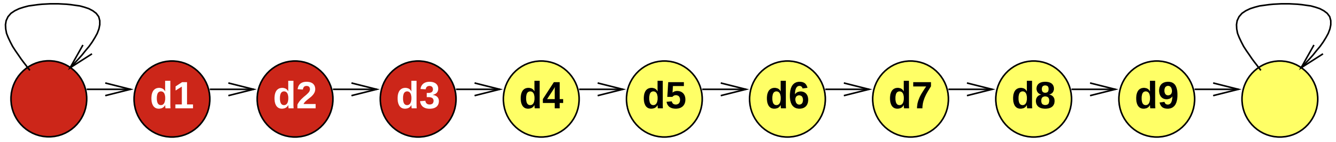




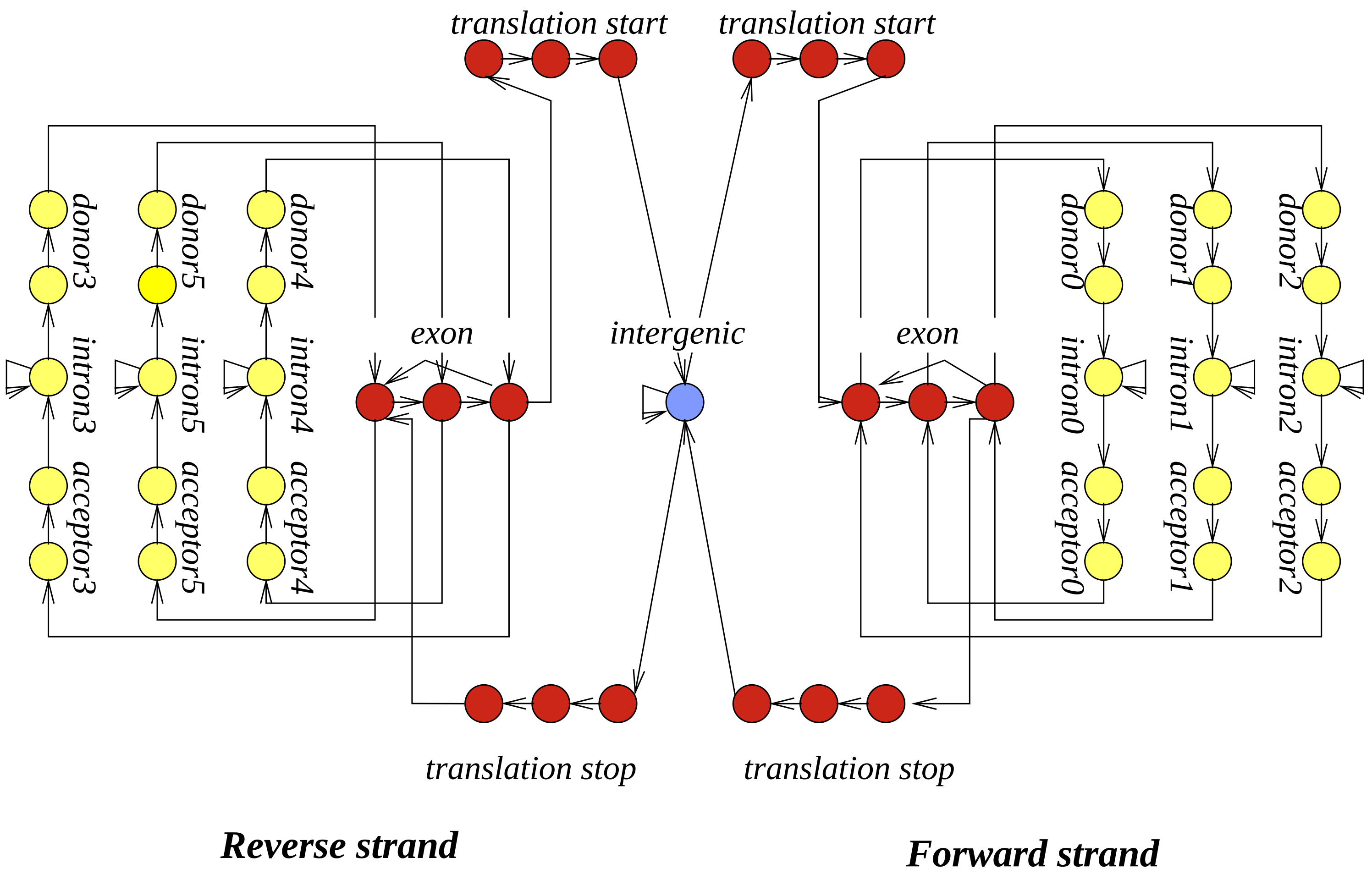
Gene finding HMM: signals

```
                Exón                            Intrón
ccatcccctatatttatggcagGTgaggaaagggtgggggctgggg
attcatcatcatgggtgcatcgGTgagtatctcccaggccccaatc
agaagatctaccccaccatctgGTaagtgtgtcccaccactgcccc
acagagtgagcccttcttcaagGTgggtggtgtcagggcctccccc
acgagtcctgcatgagccagatGTaaggcttgccgttgccctccct
tgcagaacctcatggtgctgagGTggggccaagcctgggccggggg
tcgatgaatttgggatcatccgGTgagagctcttcctctctcctgg
agatgacgtccgtgatgagaagGTagggggtgcaccccagtcccca
gtggagaatgagaggtgggatgGTaggtgatgccttcgaggcccag
tttcttgtggctattttaaaagGTaattcatggagaaatagaaaaa
```

Add a sequence of states between exon and intron:



Gene finding HMM: a complete model



Higher order states

Order 0: emission table e contains $\Pr(S_i|A_i)$

Order 1: e contains $\Pr(S_i|A_i, S_{i-1})$

A_i	S_{i-1}	a	c	g	t
🟥	a	0.24	0.23	0.34	0.19
	c	0.30	0.31	0.13	0.26
	g	0.27	0.28	0.28	0.17
	t	0.13	0.28	0.38	0.21
🟨	a	0.30	0.18	0.27	0.25
	c	0.32	0.28	0.06	0.35
	g	0.27	0.22	0.27	0.24
	t	0.20	0.21	0.26	0.33

...

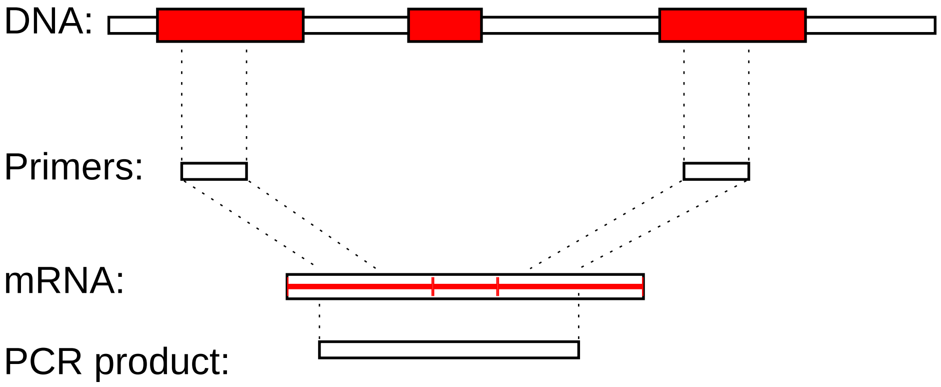
For exons, introns, etc. use orders 4-5.

Experimental verification of predicted genes

Transcription and splicing

- RNA-Seq: sequencing of all mRNAs extracted from the cell
- RT PCR: targeted verification of a specific gene using specifically designed primers

Problems: difficult to find genes that are expressed under special conditions, i.e. embryonic development
genomic DNA contamination, non-unique mapping to the genome



Experimental verification of predicted genes

Translation, existence of the protein

- Mass spectrometry
- Detection based on antibodies
- Other methods specific to individual proteins

Examples of gene finging programs

Based only on DNA sequence:

HMMGene [Krogh, 1997], Genscan [Burge and Karlin, 1997], GeneZilla [Majoros et al., 2004], ExonHunter [Brejová et al., 2005], Augustus [Stanke and Waack, 2003]

Prokaryotes:

GeneMark [Lukashin and Borodovsky, 1998], Glimmer [Delcher et al., 1999].

Examples of gene finding programs

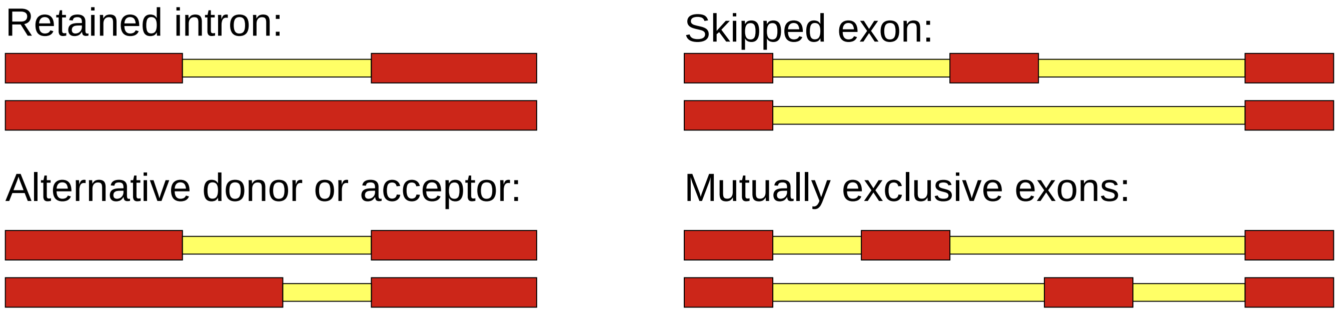
Comparison of multiple sequences:

Twinscan [Korf et al., 2001], Exoniphy [Siepel and Haussler, 2004], N-SCAN [Gross and Brent, 2006]
(Twinscan extended to multiple genomes).

Integration of additional information: (RNA-seq, proteins from related genomes, etc.)
ExonHunter [Brejová et al., 2005], Augustus [Stanke et al., 2006], Jigsaw [Allen and Salzberg, 2005], Fgenesh++ [Solovyev et al., 2006].

Limitations of gene finders

- Alternative splicing: one gene can produce different mRNAs; gene finders typically only find one



Retained intron:

Skipped exon:

Alternative donor or acceptor:

Mutually exclusive exons:

- Overlapping genes (including genes in introns)
- Atypical genes (unusual signal, short or long exons or introns)
- Untranslated regions (UTR) are difficult

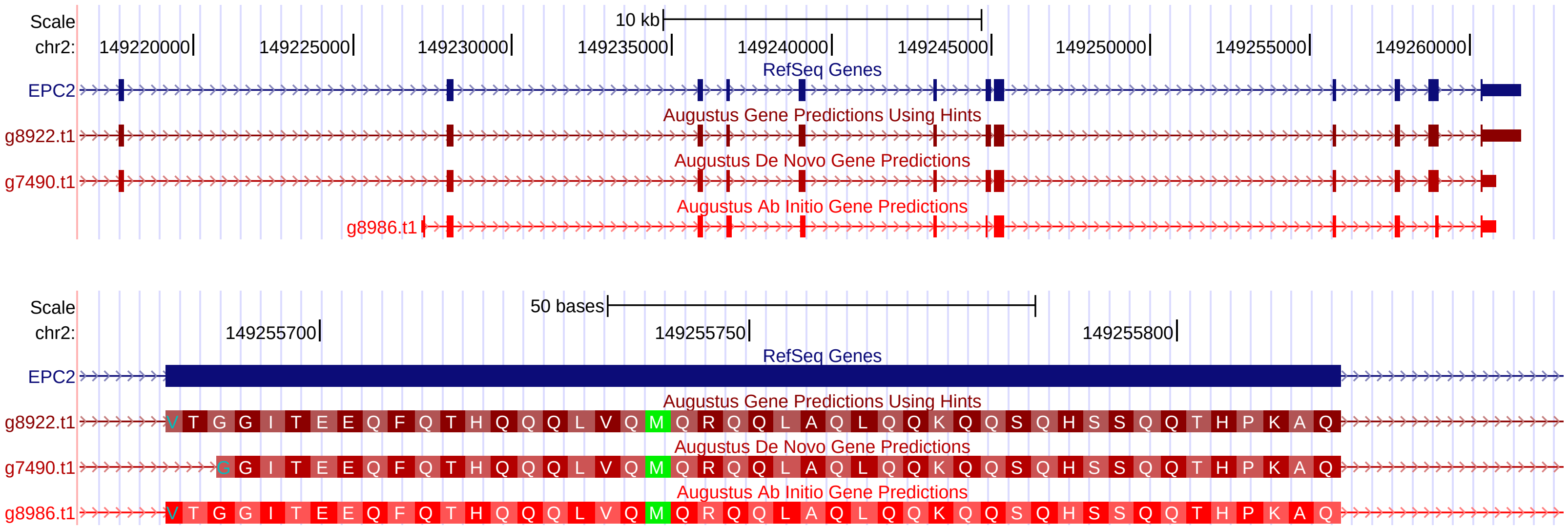
Gene finders often make errors

Results on human genome: [Guigo et al 2006]

20% genes, 60% exons correct based on DNA

35% genes, 65% exóns correct based on comparisons

70% génes, 85% exóns correct with additional info


Scale
chr2:
10 kb
149220000
149225000
149230000
149235000
149240000
149245000
149250000
149255000
149260000
RefSeq Genes
EPC2
Augustus Gene Predictions Using Hints
g8922.t1
Augustus De Novo Gene Predictions
g7490.t1
Augustus Ab Initio Gene Predictions
g8986.t1
Scale
chr2:
50 bases
149255700
149255750
149255800
RefSeq Genes
EPC2
Augustus Gene Predictions Using Hints
g8922.t1
V T G G I T E E Q F Q T H Q Q Q L V Q M Q R Q Q L A Q L Q Q K Q Q S Q H S S Q Q T H P K A Q
Augustus De Novo Gene Predictions
g7490.t1
G G I T E E Q F Q T H Q Q Q L V Q M Q R Q Q L A Q L Q Q K Q Q S Q H S S Q Q T H P K A Q
Augustus Ab Initio Gene Predictions
g8986.t1
V T G G I T E E Q F Q T H Q Q Q L V Q M Q R Q Q L A Q L Q Q K Q Q S Q H S S Q Q T H P K A Q

How many genes in human genome?

Before 2001: **50 000–140 000** genes

2001: draft human genome: **30 000–40 000** genes

2004: completed human genome: **20 000–25 000** genes

2007: Ensembl, RefSeq, VEGA catalog: **24 500** genes
[Clamp a kol. 2007] claims only **20 500** correct
Are there genes about which we don't know yet?

2010: RefSeq **22 333** genes
uncertainty of ± 1000 [Pertea, Salzberg 2010]
Individuals can differ in tens of genes

2012: Project ENCODE estimates **20 687** protein coding genes,
on average 6 transcripts per gene,
plus 8 800 short and 9 600 long RNA genes

Summary

- Newly sequenced genomes need to be annotated:
  determine functions of individual segments of the genome
- Example of annotation: finding genes that code for proteins
- Hidden Markov models are suitable for gene finding
- Models make a lot of errors, but they at least give us the basic understanding of location an number of genes, we can study their function

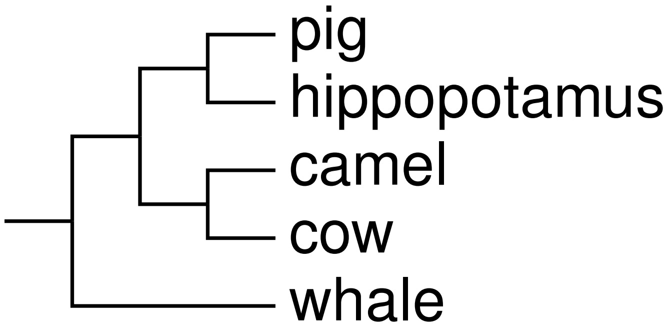
Announcements

- Homework 1 is due Tuesday November 9 22:00
  discussion regarding questions in MS Teams
- Work on the journal club
  (read the paper, plan the meeting no later than Nov. 23)

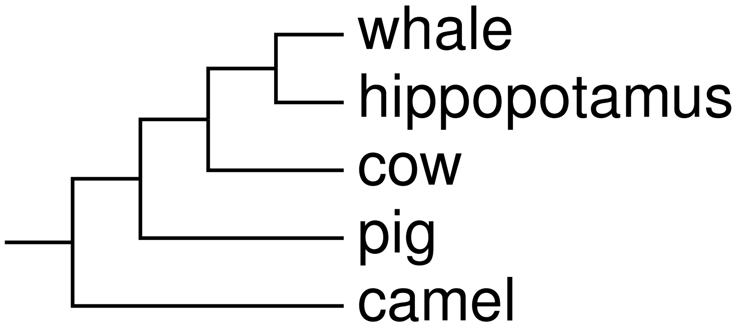
Evolution and Phylogenetic Trees

Broňa Brejová

October 28, 2021



OR



Phylogenetic tree reconstruction (fylogenetický strom)

Input:

m **aligned** sequences,

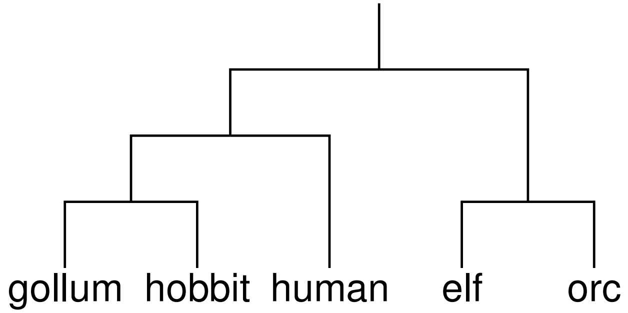
each of length n

human	C	A	G	T	T	A
elf	A	A	T	A	G	A
Gollum	C	C	G	A	G	A
hobbit	C	C	G	T	T	C
orc	A	A	T	T	T	A

Output:

tree representing

their evolutionary history

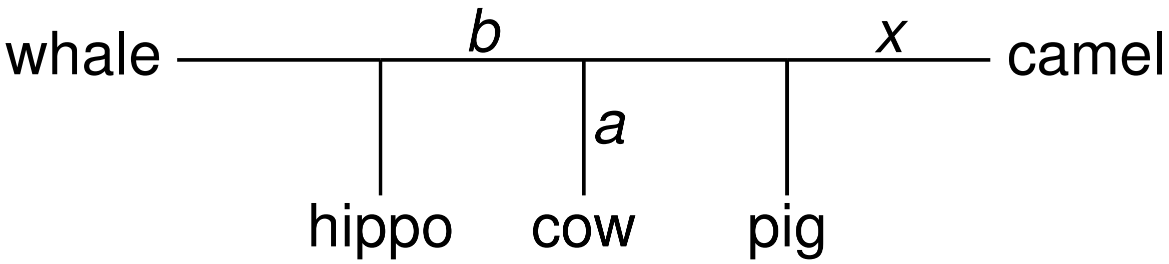


Newick format:

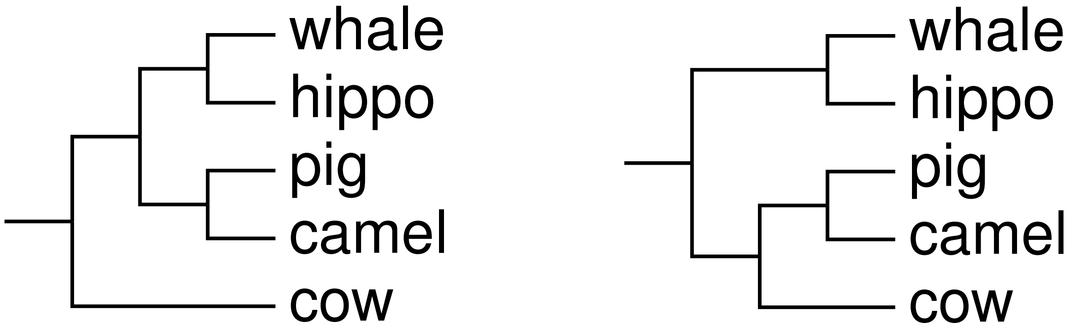
(((gollum,hobbit),human),(elf,orc))

Rooted and unrooted trees

Unrooted tree (nezakorenený strom)



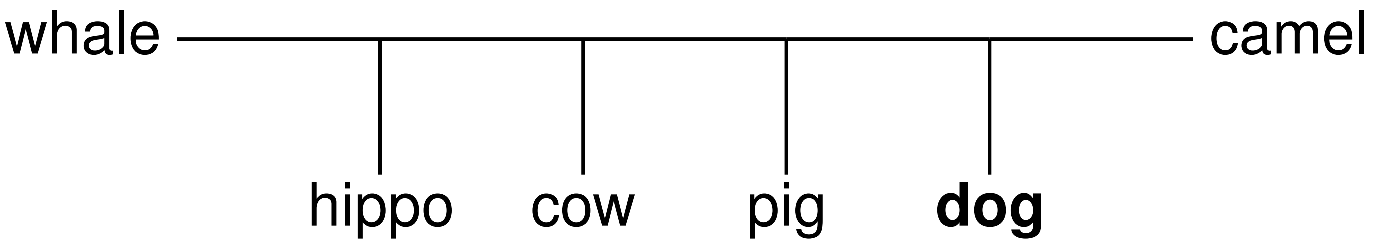
Two out of seven possible rooted versions of the tree

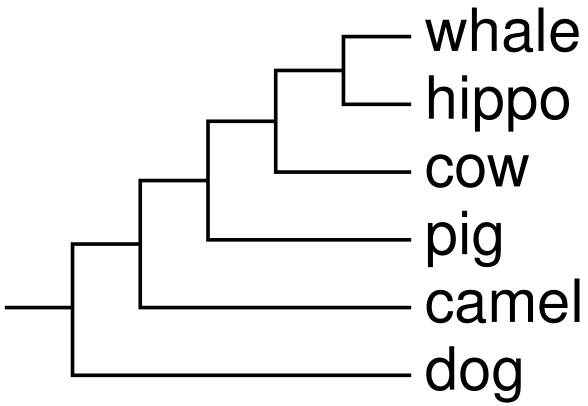


Most methods reconstruct unrooted trees

Rooting a tree using an outgroup

Add outgroup (dog) to the unrooted tree





Parsimony principle and maximum parsimony (úspornosť)

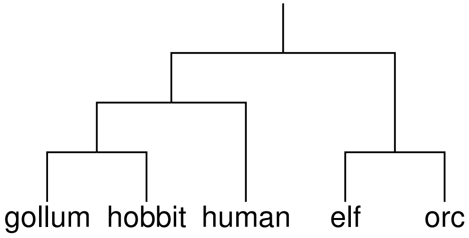
Input: (aligned) sequences of several extant species.

Task: Find a phylogenetic tree that explains the data by using the **minimum number of evolutionary events**.

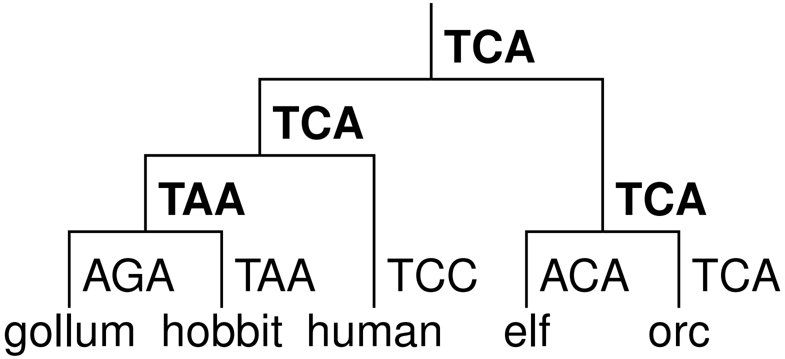
Here: Evolutionary event = single base mutation

Subtask: For a given phylogenetic tree, find **ancestral sequences** that require the minimum number of events (score of the tree)

gollum	AGA
hobbit	TAA
human	TCC
elf	ACA
orc	TCA



→



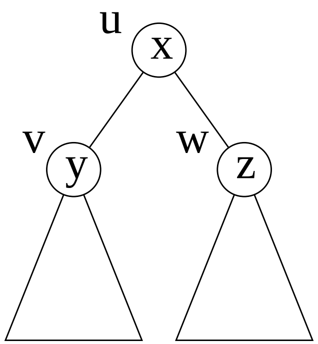
5 changes

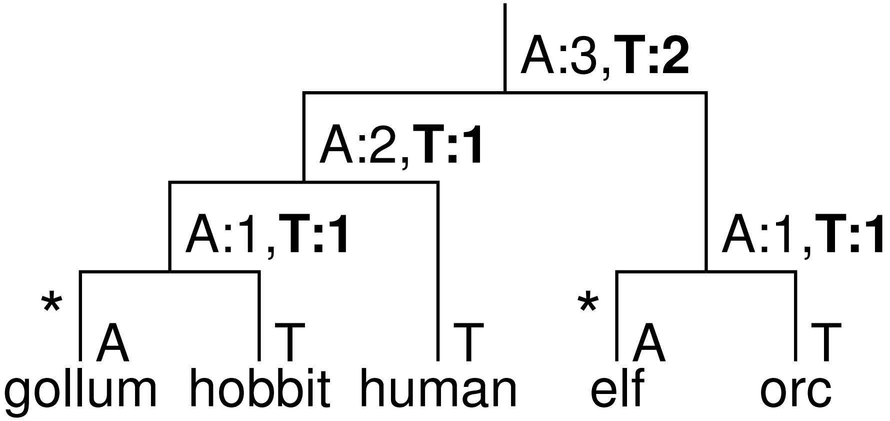
Computing cost of a given phylogenetic tree

Use **dynamic programming** (separately for each alignment column).

For each internal vertex u and symbol x:
$N_{u,x}$: how many events are required in the subtree of u, assuming that the symbol in u is x?

$$N_{u,x} = \min_y\{N_{v,y} + [x \neq y]\} + \min_z\{N_{w,z} + [x \neq z]\}$$



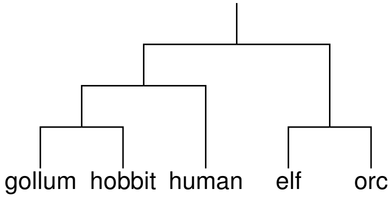


Time: $O(m)$

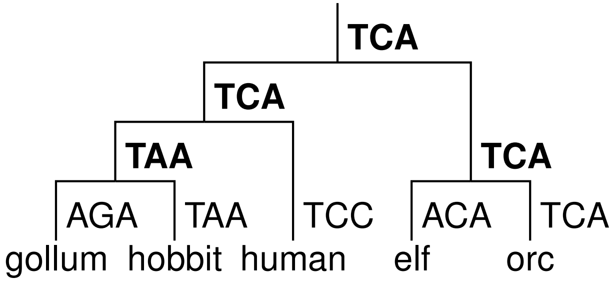
Repeat for each alignment column: $O(mn)$

What we have: compute the cost of a particular tree

gollum	AGA
hobbit	TAA
human	TCC
elf	ACA
orc	TCA



→

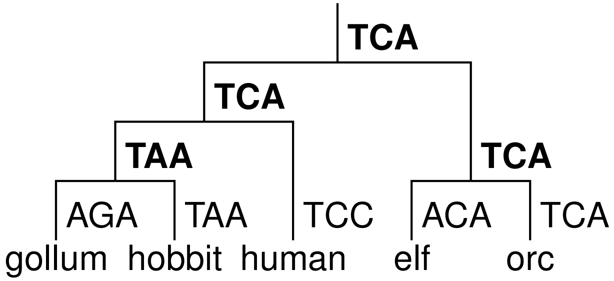


5 changes

What we want: Find the tree with the smallest cost

gollum	AGA
hobbit	TAA
human	TCC
elf	ACA
orc	TCA

→



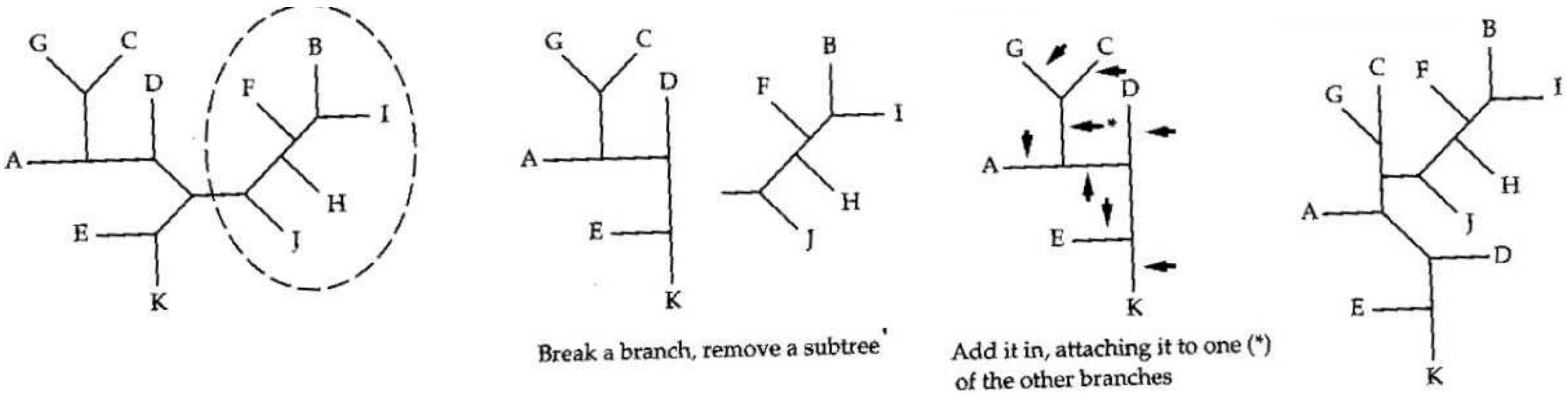
Finding the most parsimonious tree

NP-hard problem

Trivial algorithm: try all possible trees.
For m species $1 \cdot 3 \cdot 5 \cdots (2m-5) = (2m-5)!!$
E.g. for 10 species cca 2 mil., for 20 species $2 \cdot 10^{20}$

Heuristic search:

- Start with some "sensible" tree
- Explore similar trees by using e.g. "subtree pruning and regraft":



Break a branch, remove a subtree'

Add it in, attaching it to one (*) of the other branches

Neighbour joining (metóda spájania susedov)

- We throw away "details" of which mutations happened
- Summarize by a **distance matrix** D_{ij}

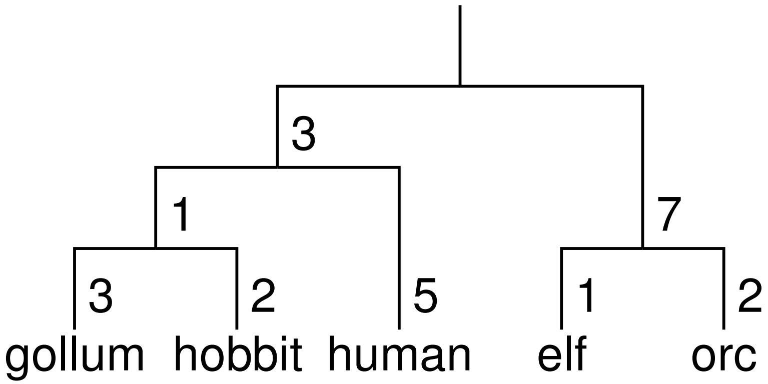
Example:

human	C	A	G	T	T	A
elf	A	A	T	A	G	A
gollum	C	C	G	A	G	A
hobbit	C	C	G	T	T	C
orc	A	A	T	T	T	A

	hu	e	h	ho	o
human	0	4	3	2	2
elf	4	0	3	6	2
gollum	3	3	0	3	5
hobbit	2	6	3	0	4
orc	2	2	5	4	0

Idea of neighbour joining

Assume that the distances $D_{i,j}$ correspond to the real distances in the tree (they are **additive**)



	gollum	hobbit	human	elf	orc
gollum	0	5	9	15	16
hobbit	5	0	8	14	15
human	9	8	0	16	17
elf	15	14	16	0	3
orc	16	15	17	3	0

$D_{\text{hobbit,human}} = 2 + 1 + 5 = 8$

Idea of neighbour joining

- Assume that the distances $D_{i,j}$ correspond to the real distances in the tree (they are **additive**)

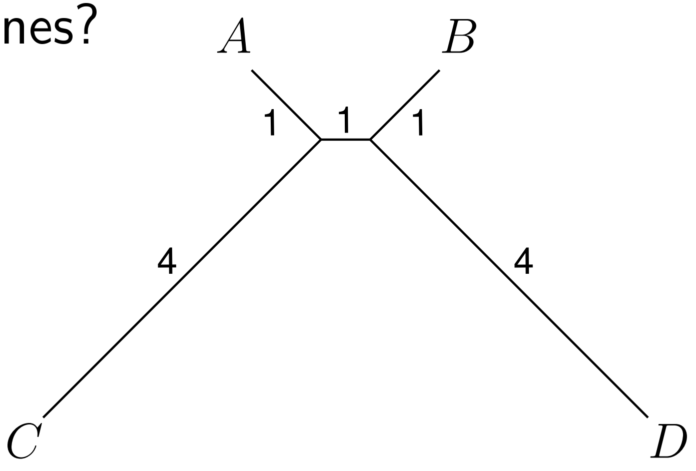
- Find two leaves i and j, for which we can **say with certainty**, that they have the same parent in the tree

- Join i and j and replace them with a parent node k with new distances to each other node ℓ:

$$D_{k,\ell} = \frac{D_{i,\ell} + D_{j,\ell} - D_{i,j}}{2}$$

How to find out which two leaves should be joined?

Why not two closest ones?



	A	B	C	D
A	-	3	5	6
B	3	-	6	5
C	5	6	-	9
D	6	5	9	-

Choose leaves i, j **minimizing**:

$$L_{i,j} = (m-2)D_{i,j} - \underbrace{\sum_{k \neq i} D_{i,k}}_{r_i} - \underbrace{\sum_{k \neq j} D_{j,k}}_{r_j}$$

m: the number of leaves

Connect leaves i, j, which minimize the following quantity:

$$L_{i,j} = (m-2)D_{i,j} - \underbrace{\sum_{k \neq i} D_{i,k}}_{r_i} - \underbrace{\sum_{k \neq j} D_{j,k}}_{r_j}$$

D

	g	ho	hu	e	o	r_i
g	0	5	9	15	16	45
ho	5	0	8	14	15	42
hu	9	8	0	16	17	50
e	15	14	16	0	3	48
o	16	15	17	3	0	51

L

	g	ho	hu	e	o
g	.	-72	-68	-58	-48
ho	-72	.	-68	-48	-48
hu	-68	-68	.	-50	-50
e	-58	-48	-50	.	**-90**
o	-48	-48	-50	**-90**	.

new D

	g	ho	hu	e+o
g	0	5	9	14
ho	5	0	8	13
hu	9	8	0	15
e+o	14	13	15	0

Running time of neighbor joining: $O(m^3)$ (m: number of leaves)

In 2009 a $O(m^2)$ version was developed (Elias and Lagergren)

Neighbour joining: summary

- If the distance matrix is additive and corresponds to the real evolutionary distances then neighbour joining finds the correct tree
- Longer sequences ⇒ better distance estimates ⇒ correct trees
- How to compute “real” evolutionary distances?
  Counting differences is not enough

human	C	A	G	T	T	A
elf	A	A	T	A	G	A
gollum	C	C	G	A	G	A
hobbit	C	C	G	T	T	C
orc	A	A	T	T	T	A

	hu	e	g	ho	o
human	0	4	3	2	2
elf	4	0	3	6	2
gollum	3	3	0	3	5
hobbit	2	6	3	0	4
orc	2	2	5	4	0

Problems with estimating distances

- One base may mutate multiple times during evolution (possibly even back to original base)
- When counting differences we see at most one change at each position $\Rightarrow$ we underestimate the real distance
- We want a correction to estimate the real number of mutations that have occurred

Jukes-Cantor substitution model

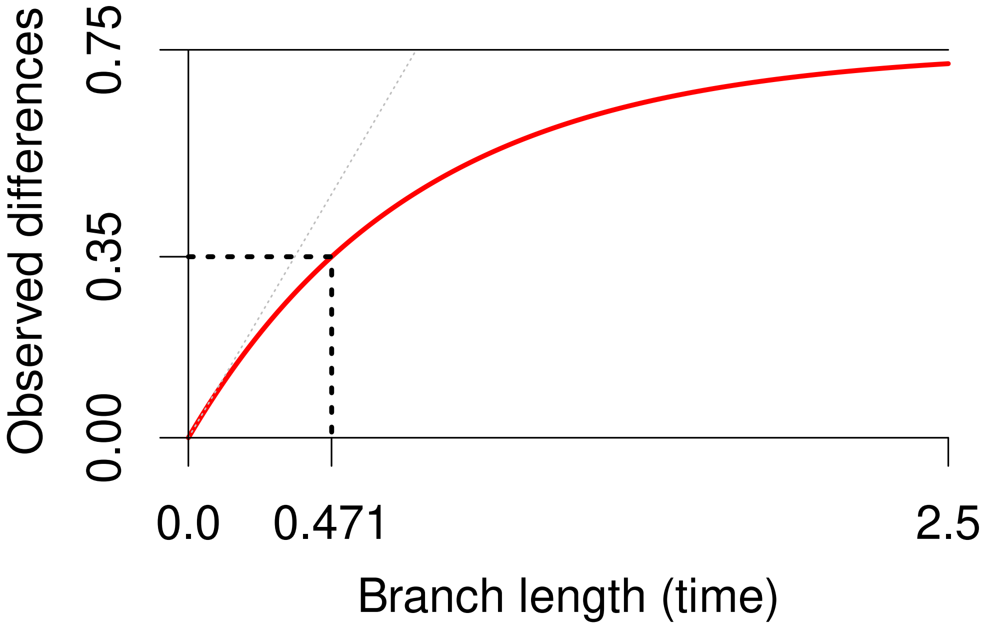
Probability that base A changes to C in time t:

$$\Pr(X_{t_0+t} = C \,|\, X_{t_0} = A) = \tfrac{1}{4}(1 - e^{-\frac{4}{3}\alpha t})$$

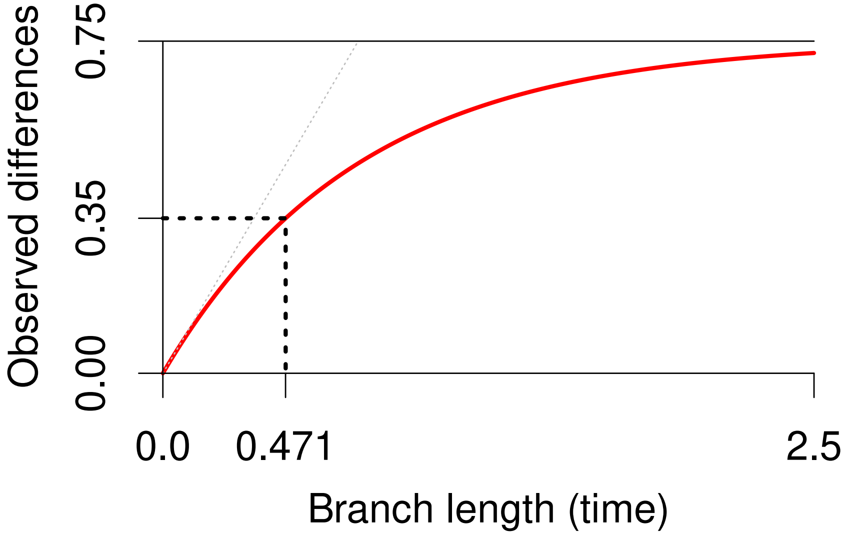
α: mutation rate (the number of substitutions per unit of time)

Expected number of observed changes per base in time t:

$$D(t) = \Pr(X_{t_0+t} \neq X_{t_0}) = \tfrac{3}{4}(1 - e^{-\frac{4}{3}\alpha t})$$



Back to distances in neighbor joining



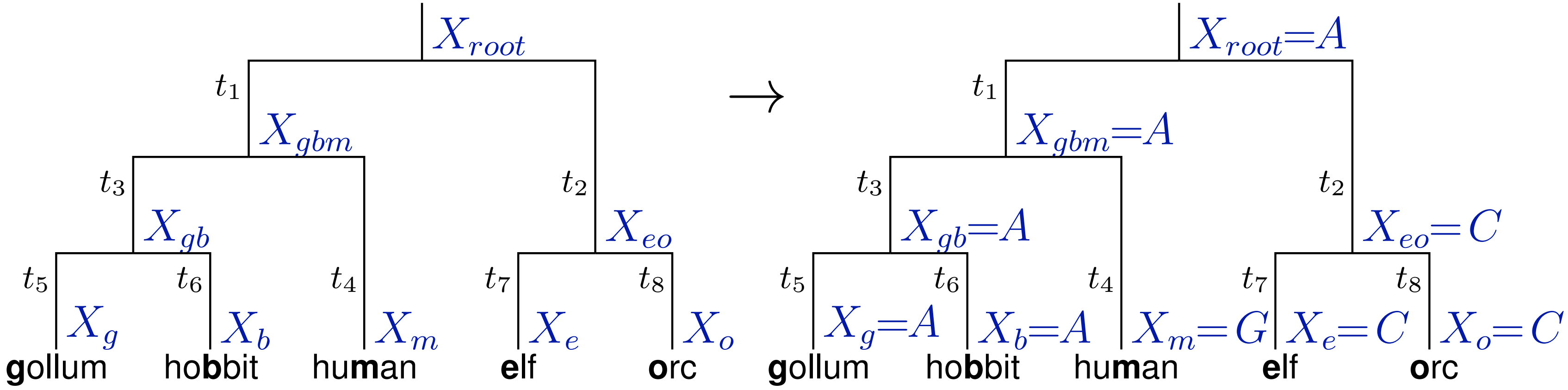
- Using this model, we can correct observed distances

$$D = \frac{3}{4}(1 - e^{-\frac{4}{3}\alpha t}) \qquad \Rightarrow \qquad \alpha t = -\frac{3}{4}\ln(1 - \frac{4}{3}D)$$

- Next week: more complex models of evolution

Maximum likelihood trees (najvierohodnejšie stromy)

A phylogenetic tree with branch lengths can be viewed as a **simple generative model**



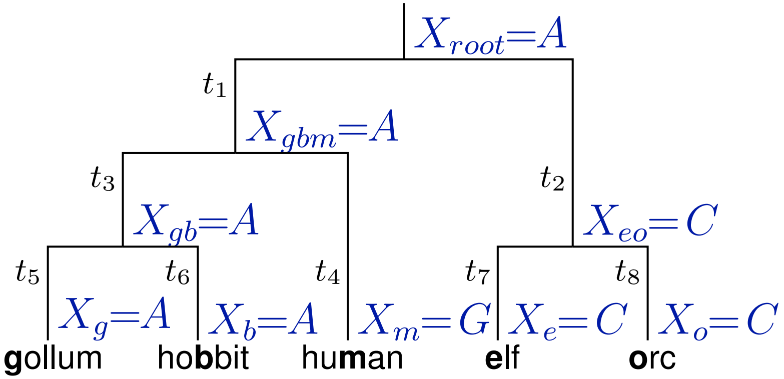
Probability that it generates particular bases in nodes:

$$\Pr(X_g=A, X_b=A, X_m=G, X_e=C, X_o=C, X_{gb}=A,$$
$$X_{gbm}=A, X_{eo}=C, X_{root}=A)$$
$$= \Pr(X_{root}=A)\cdot\Pr(A\,|\,A,t_1)\cdot\Pr(C\,|\,A,t_2)\cdot\Pr(A\,|\,A,t_3)\cdot$$
$$\Pr(G\,|\,A,t_4)\cdot\Pr(A\,|\,A,t_5)\cdot\Pr(A\,|\,A,t_6)\cdot\Pr(C\,|\,C,t_7)\cdot\Pr(C\,|\,C,t_8)$$

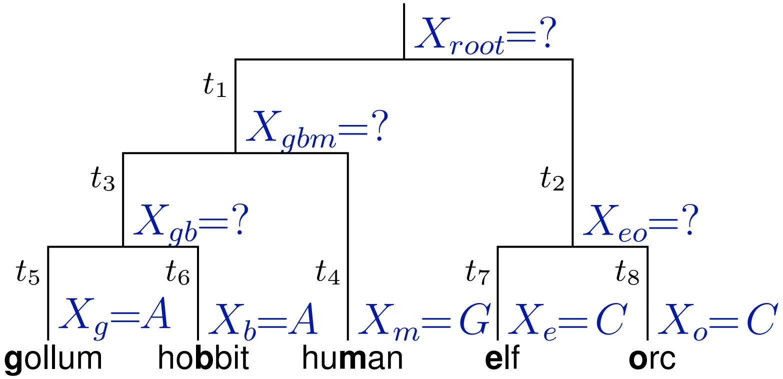
$\Pr(C|A,t_2)$ is a abbreviation of $\Pr(X_{eo}=C|X_{root}=A)$, J.-C. model

We can compute (product):

We want to compute **tree likelihood**:



$X_{root}=A$
t_1
$X_{gbm}=A$
t_3
t_2
$X_{gb}=A$
$X_{eo}=C$
t_5 t_6 t_4 t_7 t_8
$X_g=A$ $X_b=A$ $X_m=G$ $X_e=C$ $X_o=C$
gollum hobbit human elf orc



$X_{root}=?$
t_1
$X_{gbm}=?$
t_3
t_2
$X_{gb}=?$
$X_{eo}=?$
t_5 t_6 t_4 t_7 t_8
$X_g=A$ $X_b=A$ $X_m=G$ $X_e=C$ $X_o=C$
gollum hobbit human elf orc

Likelihood of a tree (vierohodnosť stromu):

$\Pr(X_g=A, X_b=A, X_m=G, X_e=C, X_o=C)$

Add up probabilities of all letter combinations in ancestors X_{gb}, X_{gbm}, X_{eo}, X_{root}

Compute using **Felsenstein algorithm**

(simple dynamic programming similar to the parsimony)

For a given alignment, tree and branch lengths
we can compute likelihood in $O(nm)$ time

How to find the tree with the highest likelihood?

- Again NP-hard problem ;
  complicated because we also need **branch lengths**
- Typical heuristic algorithm:
  - Start with a “reasonable” tree
  - Compute its likelihood
    - Start with “reasonable” branch lengths
    - Compute likelihood using these branch lengths
    - Iteratively improve branch lengths to improve the likelihood (e.g. gradient descent)
  - Explore “similar” trees to improve likelihood (as with parsimony).

Consistency of algorithms for phylogeny

- "Well-behaved" algorithms: if the length of the sequences n increases, the answer should get closer to the correct answer.
- The algorithm for phylogeny is **consistent**, if the probability of obtaining the correct tree converges to 1 with $n \to \infty$.

Algorithm comparison

	Complexity	**Consistency**	**Data utilization**
Parsimony	NP-hard	NO	complete sequences
Neighbor Joining	$O(m^3)$	YES	distances only
Likelihood	NP-hard	YES	complete sequences

Sources of data for phylogenetic trees

Some special sequences are often used
(e.g. ribosomal RNA genes, mitochondrial genome)

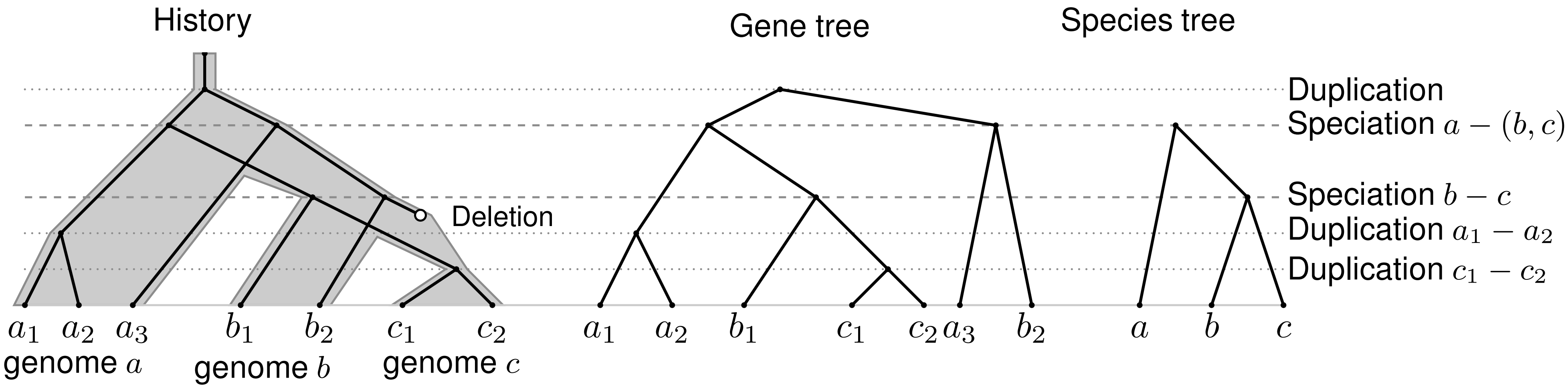
What about using DNA sequences of other genes?

- Choose a suitable gene
- Find its homologs in other species
- Use these to construct the tree
  (DNA sequences or proteins)

Problem: genes can be duplicated and lost in evolution

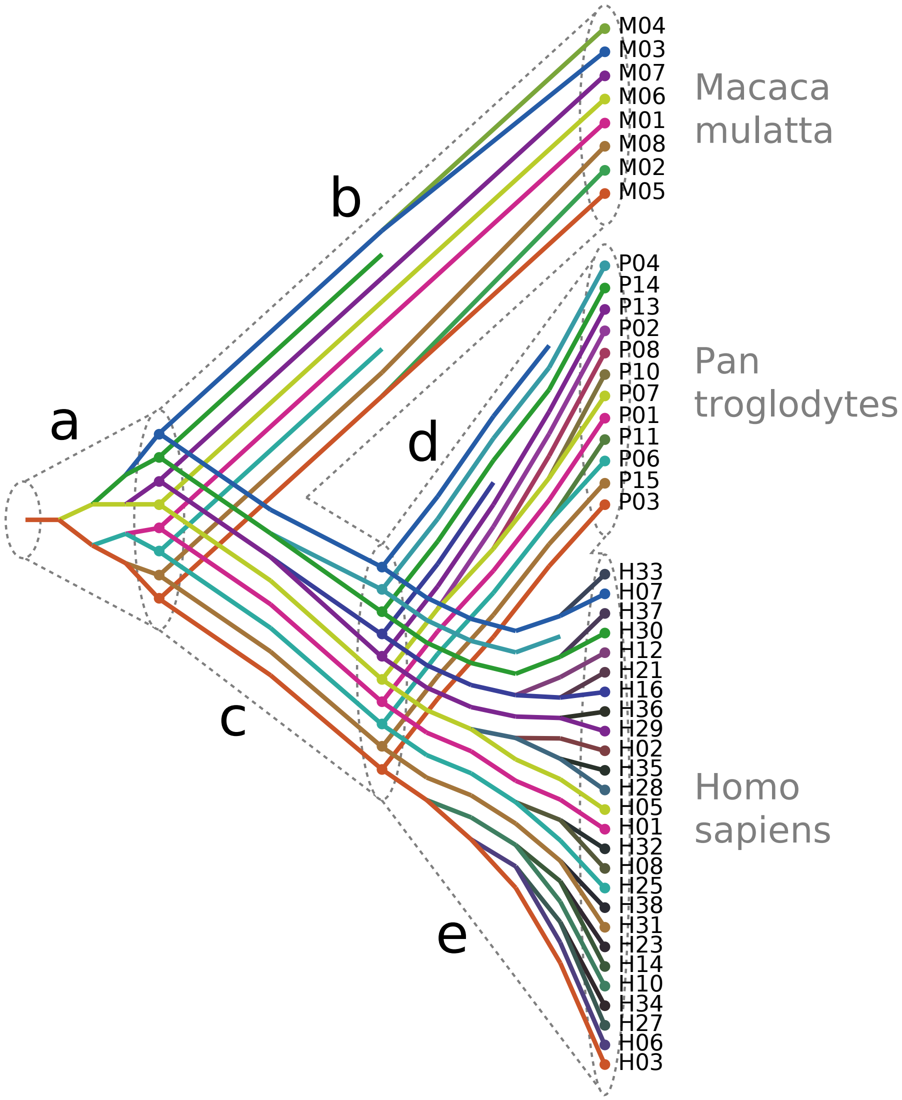
History of a duplicated gene

Example: species a, b, c, genes $a_1, a_2, a_3, b_1, b_2, c_1, c_2$



- **Homologs:** similar sequences evolved from a common ancestor
- **Orthologs:** closest common ancestor is a speciation
  (e.g. pairs of genes $a_1 - b_1$, $a_2 - b_1$)
- **Paralogs:** closest common ancestor is a duplication
  (e.g. pairs of genes $a_1 - a_2$, $a_1 - b_2$)

A more complex example of gene duplication:



Summary

Substitution models allow us to:

- estimate real evolutionary distance (the number of substitutions) from the observed difference count between two sequences
- compute the probability that we observe a particular nucleotide change over time t

Three methods for phylogeny inference:

- Parsimony
- Neighbour joining
- Maximum likelihood

Gene trees and species trees, complications in phylogeny reconstruction

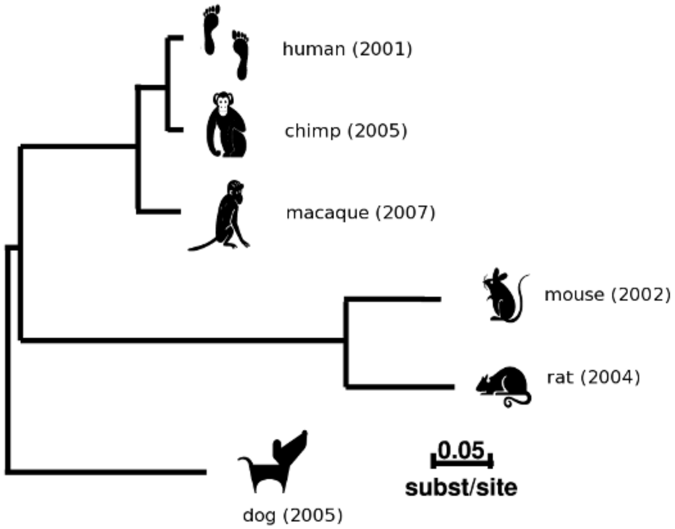
Announcements

- Homework 1 is due next Tuesday, November 9 22:00
  submit in Moodle, guests by email to brejovadcs.fmph.uniba.sk
  discussion regarding questions in MS Teams
- Work on the journal club
  (read the paper, plan the meeting no later than Nov. 23)
- Next week Bratislava in the red zone
  we will try to keep the possibility of in-person classes

Comparative Genomics

Tomáš Vinař

November 4, 2021



Comparative genomics (komparatívna genomika)

- Genome evolution:
  - Single point mutations (this lecture)
  - Short insertions and deletions
  - Large-scale events: rearrangements and duplications
- Mutations according to their effect:
  - Neutral
  - Deleterious (škodlivé)
    ⇒ **purifying selection (purifikačný výber)**
  - Advantageous (prospešné)
    ⇒ **positive selection (pozitívny výber)**
- By comparing several genomes,
  find regions that evolve in an unusual way
  (e.g. conserving an important function, evolving a new function)

Comparative genomics

- Start with multiple alignment of several genomes (aligned sites should have originated from the same ancestral sequence)

```
         Human AGTGGCTGCCAGGCTG---GGATGCTGAGGCCTTGTTTGCAGGGAGGT
        Rhesus AGTGGCTGCCAGGCTG---GGTTGCTGAGGCCTTGTTTGCCGGGAGGT
         Mouse GGTGGCTGCCGGGCTG---GGTGGCTGAGGCCTTGTTGGTGGGGTGGT
           Dog AGTGGCTGCCCGGCTG---GGTGGCTGAGGCCTTATTTGCAGGGAGGT
         Horse GATGGCTGCCGGGCTG---GGCTGCCGAGGCCTTGTTCGTGGGGAGGT
     Armadillo AGTGGCTGCCGGGCTG---GGAGGCCAAGGCCTTGTTCGCGGGCAGGT
       Chicken AGTGGCTGCCAGTCTGCGCCGTGGCCGACGTCTTGCTCGGGGGAAGGT
X. tropicalis AATGGCTTCCATTTTGTGCCGCTGCTGAGGTCTTGTTCTGGGGAAGAT
```

- **Methods:** Combine techniques for sequence annotation (HMMs) and evolutionary models

Application 1: Finding functional elements of the genomes

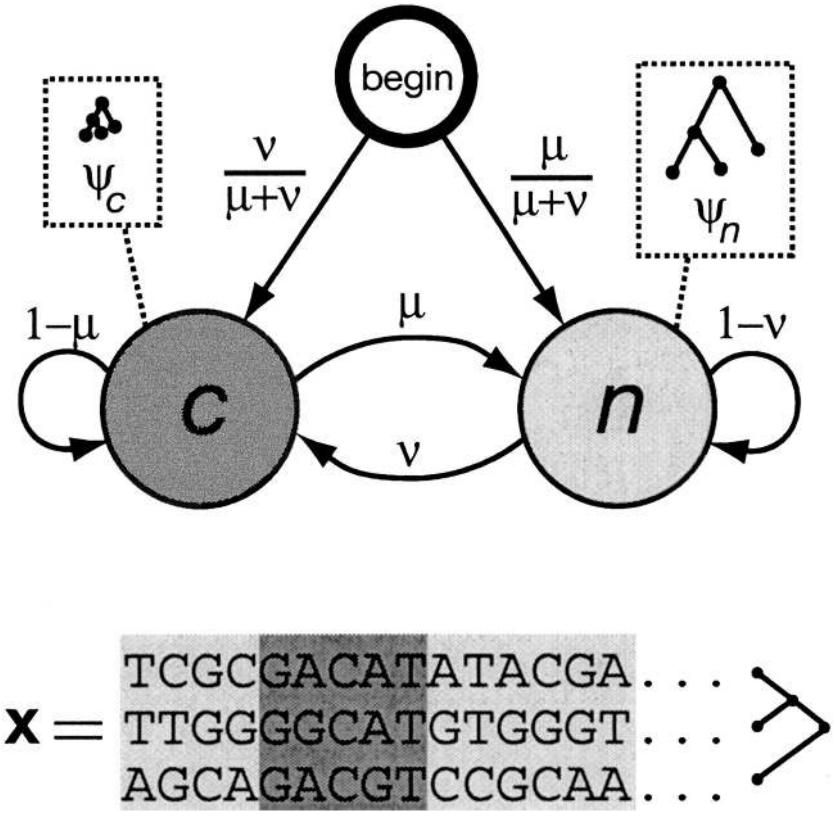
Consequences of purifying selection

- Important functional sequences are likely to be conserved: they appear to evolve slower
- Non-functional sequences evolve faster
- **Example:** protein coding genes in humans and mouse
  - coding regions: 85% identity (98% of their total length aligned)
  - introns: 69% identity (48% of their total length aligned)
- **Task:** find **well-conserved sequences** between organisms
- Majority of conserved sequences will correspond to known functional elements (coding genes, regulation sequences, etc.)
- Conserved sequences that do not overlap known functional elements: interesting objects for further research

PhastCons: detection of conserved sequences

Phylogenetic HMM:

combination of an HMM and a phylogenetic tree



Source: [Siepel et al., 2005]

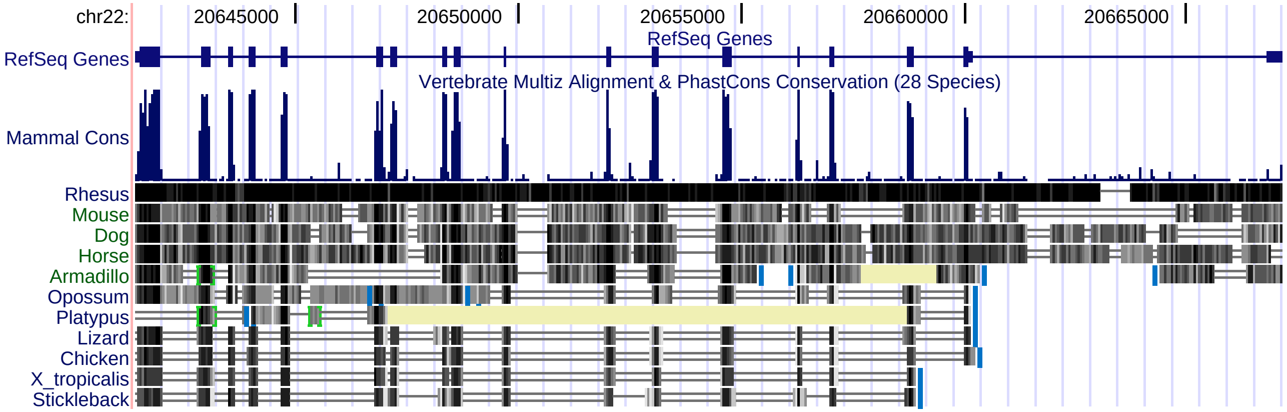
- Two states: conserved and neutral
- Each state emits a whole column of a sequence alignment
- Conserved sequences have shorter tree branches, causing less sequence divergence

How to use phylogenetic HMMs

- The model gives a probability distribution over all possible alignments and annotations
  (here: annotation = markup of conserved / neutral regions)

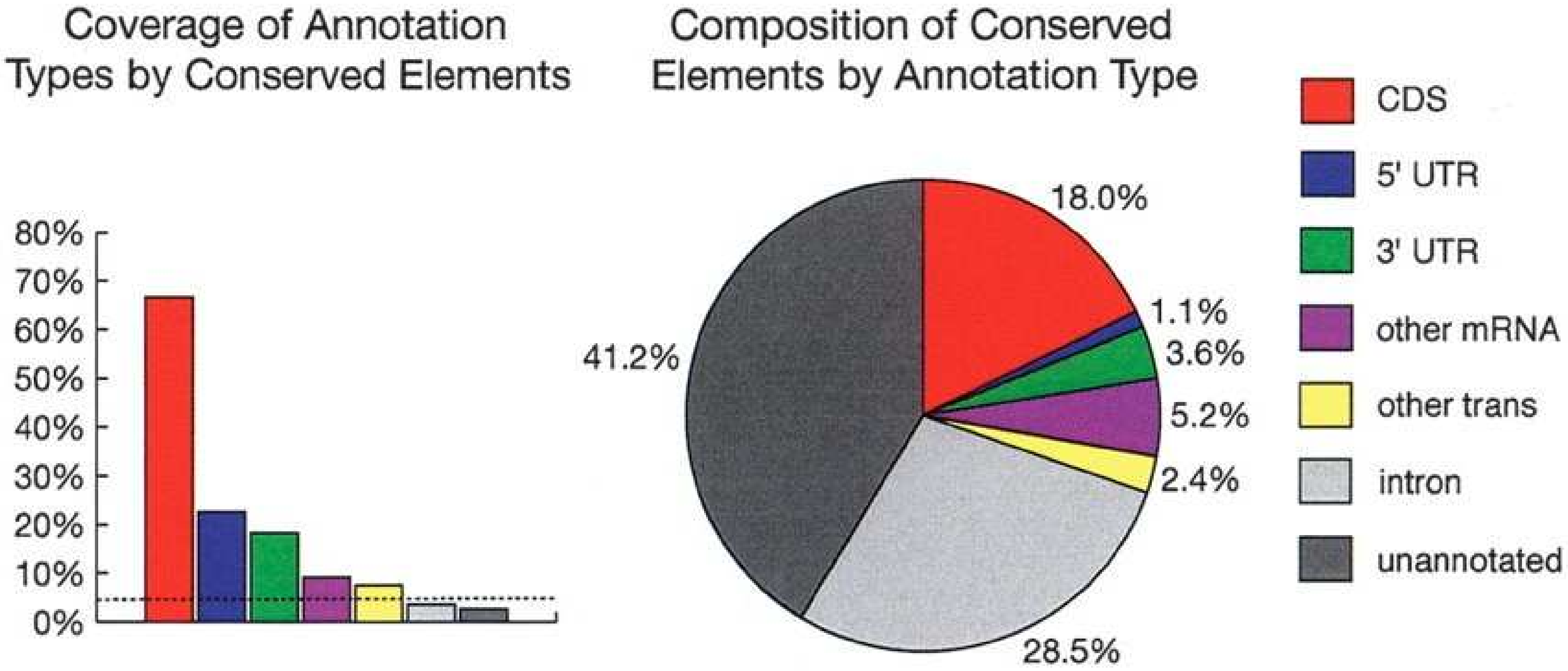
- For a given alignment, we are looking for the annotation that would maximize this probability

- Can be done efficiently
  (combination of the Viterbi and Felsenstein algorithms)



Results of PhastCons application to four whole genomes

Alignment of human, mouse, chicken, fugu



Source: [Siepel et al., 2005]

Phylogenetic HMMs for gene finding

- Use states from a typical gene finder
- Each state has a separate evolutionary model (rate matrix, branch lengths)
- Mutation frequencies in coding regions are three-periodic; this helps to find genes

How much we can improve on gene finding results?

	Exons		Genes	
Program	sn	sp	sn	sp
AUGUSTUS (1 genome)	52%	63%	24%	17%
NSCAN (alignment)	68%	82%	35%	37%

Guigo et al 2006, 1% of the human genome

Genetic code

Ala / A	GCT, GCC, GCA, GCG	Leu / L	TTA, TTG, CTT, CTC, CTA, CTG
Arg / R	CGT, CGC, CGA, CGG, AGA, AGG	Lys / K	AAA, AAG
Asn / N	AAT, AAC	Met / M	ATG
Asp / D	GAT, GAC	Phe / F	TTT, TTC
Cys / C	TGT, TGC	Pro / P	CCT, CCC, CCA, CCG
Gln / Q	CAA, CAG	Ser / S	TCT, TCC, TCA, TCG, AGT, AGC
Glu / E	GAA, GAG	Thr / T	ACT, ACC, ACA, ACG
Gly / G	GGT, GGC, GGA, GGG	Trp / W	TGG
His / H	CAT, CAC	Tyr / Y	TAT, TAC
Ile / I	ATT, ATC, ATA	Val / V	GTT, GTC, GTA, GTG
START	ATG	STOP	TAA, TGA, TAG

Application 2: Detecting positive selection in protein coding genes

- **Positive selection:** process that helps to fix **advantageous mutations** in a genome
- Unusually high number of mutations that can lead to change of function
- Mutations in protein coding genes:
  - Synonymous: do not change encoded amino acid
    e.g. ACA (Thr) → ACT (Thr)
  - Nonsynonymous: change the amino acid
    e.g. ACA (Thr) → AAA (Lys)
- We create a probabilistic model of evolution distinguishing synonymous and nonsynonymous mutations ⇒ identification of sequences with unusually high fraction of nonsynonymous mutations

From Jukes-Cantor to more general substitution models

- Jukes-Cantor assumes all mutations are equally probable
- In general μ_{xy} is the substitution rate from base x to base y
- **Substitution rate matrix** (matica rýchlostí)

$$\begin{pmatrix} -\mu_A & \mu_{AC} & \mu_{AG} & \mu_{AT} \\ \mu_{CA} & -\mu_C & \mu_{CG} & \mu_{CT} \\ \mu_{GA} & \mu_{GC} & -\mu_G & \mu_{GT} \\ \mu_{TA} & \mu_{TC} & \mu_{TG} & -\mu_T \end{pmatrix}$$

$$\begin{pmatrix} -\mu_A & \mu_{AC} & \mu_{AG} & \mu_{AT} \\ \mu_{CA} & -\mu_C & \mu_{CG} & \mu_{CT} \\ \mu_{GA} & \mu_{GC} & -\mu_G & \mu_{GT} \\ \mu_{TA} & \mu_{TC} & \mu_{TG} & -\mu_T \end{pmatrix}$$

For given time interval t, we can compute probability of each possible substitution (**transition probabilities**):

$$\Pr(X = C \mid Y = A, t)$$

Decreasing the number of parameters — HKY model

Hasegawa, Kishino and Yano [Hasegawa et al., 1985]

$$\begin{pmatrix} -\mu_A & \beta\pi_C & \alpha\pi_G & \beta\pi_T \\ \beta\pi_A & -\mu_C & \beta\pi_G & \alpha\pi_T \\ \alpha\pi_A & \beta\pi_C & -\mu_G & \beta\pi_T \\ \beta\pi_A & \alpha\pi_C & \beta\pi_G & -\mu_T \end{pmatrix} \qquad \mu_{x,y} = \begin{cases} \alpha\pi_y & \text{if } x \Leftrightarrow y \text{ is transition} \\ \beta\pi_y & \text{if } x \Leftrightarrow y \text{ is transversion} \end{cases}$$

- frequencies $\pi_A, \pi_C, \pi_G, \pi_T$
  (equilibrium, do not change over time)
- **transition rate (rýchlosť tranzícií)** α: $C \Leftrightarrow T, A \Leftrightarrow G$
- **transversion rate (rýchlosť tranzverzií)** β: $\{C, T\} \Leftrightarrow \{A, G\}$
- Only four parameters: $\pi_A, \pi_C, \pi_G, \kappa = \alpha/\beta$

Codon substitution models

Rate matrices on **codons** rather than single nucleotides

Rate of substitution from codon i to codon j:

$$\mu_{i,j} = \begin{cases} 0, & \text{if } i,j \text{ differ at } > 1 \text{ positions,} \\ \alpha\pi_j, & \text{synonymous transitions,} \\ \beta\pi_j, & \text{synonymous transversions,} \\ \omega\alpha\pi_j, & \text{nonsynonymous transitions,} \\ \omega\beta\pi_j, & \text{nonsynonymous transversions.} \end{cases}$$

Example: $\mu_{AAC,GGC} = 0$, $\mu_{CTA,CTT} = \beta\pi_{CTT}$, $\mu_{CTA,CCA} = \omega\alpha\pi_{CCA}$

Parameters: Codon frequencies π_j, ω, $\kappa = \alpha/\beta$

Selection: neutral evolution $\omega = 1$, positive selection $\omega > 1$, purifying selection $\omega < 1$

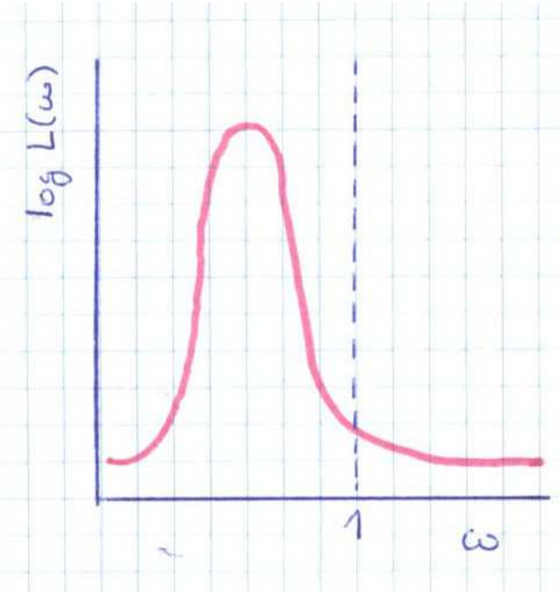
Application of codon substitution models

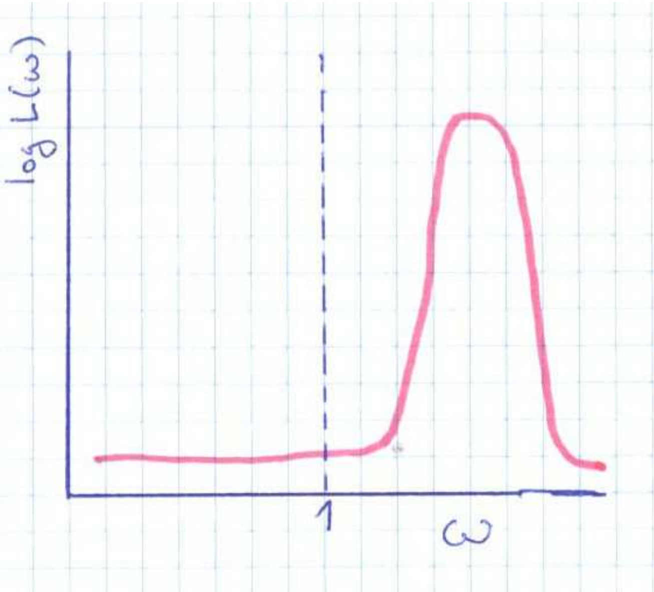
	F	V	I	H	D	S	E	G	D	G	E	C	M	Q	E
human	TTT	GTG	ATC	CAC	GAC	TCC	GAG	GGG	GAC	GGC	GAG	TGC	ATG	CAG	GAG
marmoset	TTT	GTG	ATC	CAC	GAG	AAC	AAC	AAG	GAC	GGC	GAG	TGC	ATG	CAG	GAT
	F	V	I	H	E	N	N	K	D	G	E	C	M	Q	D

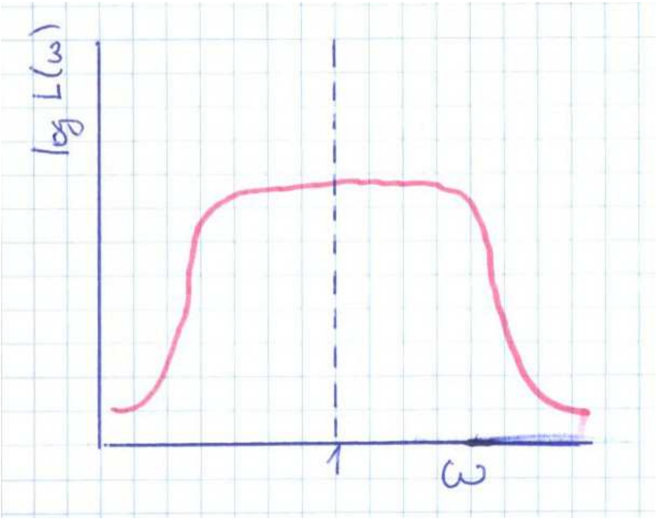
- Using whole genomes, estimate basic model parameters $\pi_A, \pi_C, \pi_G, \pi_T, \kappa$
- For a given ω and t, we can compute likelihood (vierohodnosť)

$$L(\omega, t) = \Pr(H, M \mid \omega, t)$$

- We can observe how $L(\omega) = \max_t L(\omega, t)$ changes for different values of ω


log L(ω)
1
ω


log L(ω)
1
ω


log L(ω)
1
ω

Likelihood-ratio test (test pomerov vierohodností)

- Even if $L(\omega)$ achieves maximum for $\omega > 1$,
  this can be caused by a statistical variantion in the data
  $\Rightarrow$ we need a statistical test
- Compute likelihood $L_A = \max_{\omega<1} L(\omega)$
- Compute likelihood $L_B = \max_{\omega} L(\omega)$ (no restriction on ω)
- Always $L_B \geq L_A$
- If real $\omega < 1$, then $L_A \approx L_B$ (null hypothesis)
  we are interested in cases $L_B >> L_A$
  $\Rightarrow$ the gene is under positive selection (alt. hypothesis)

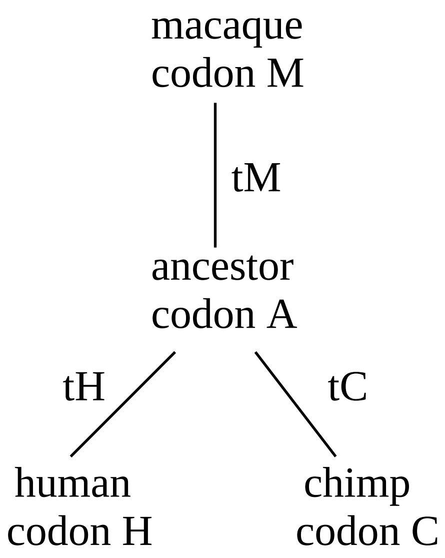
Assuming $\omega < 1$, we have $2\log(L_B/L_A) \approx \chi^2_1$
$\Rightarrow$ we can assign P-value to the null hypothesis $\omega < 1$

Detecting positive selection in protein coding genes (summary)

- Align sequences of the same gene from two species (at the codon level)
- Estimate basic parameters of the codon model using whole genome data
- Parameter ω models selection
- Compute likelihoods $L_A = \max_{\omega<1} L(\omega)$ and $L_B = \max_{\omega} L(\omega)$
- Using statistics $2\log(L_B/L_A)$, assign P-value to the null hypothesis $\omega < 1$
- Genes with small P-values are under the positive selection

“Simple” extension to multiple genomes

$$\Pr(A,H,C,M\,|\,\omega,t_H,t_C,t_M) =$$
$$\pi_A \cdot \Pr(H\,|\,A,t_H) \cdot \Pr(C\,|\,A,t_C) \cdot \Pr(M\,|\,A,t_M)$$



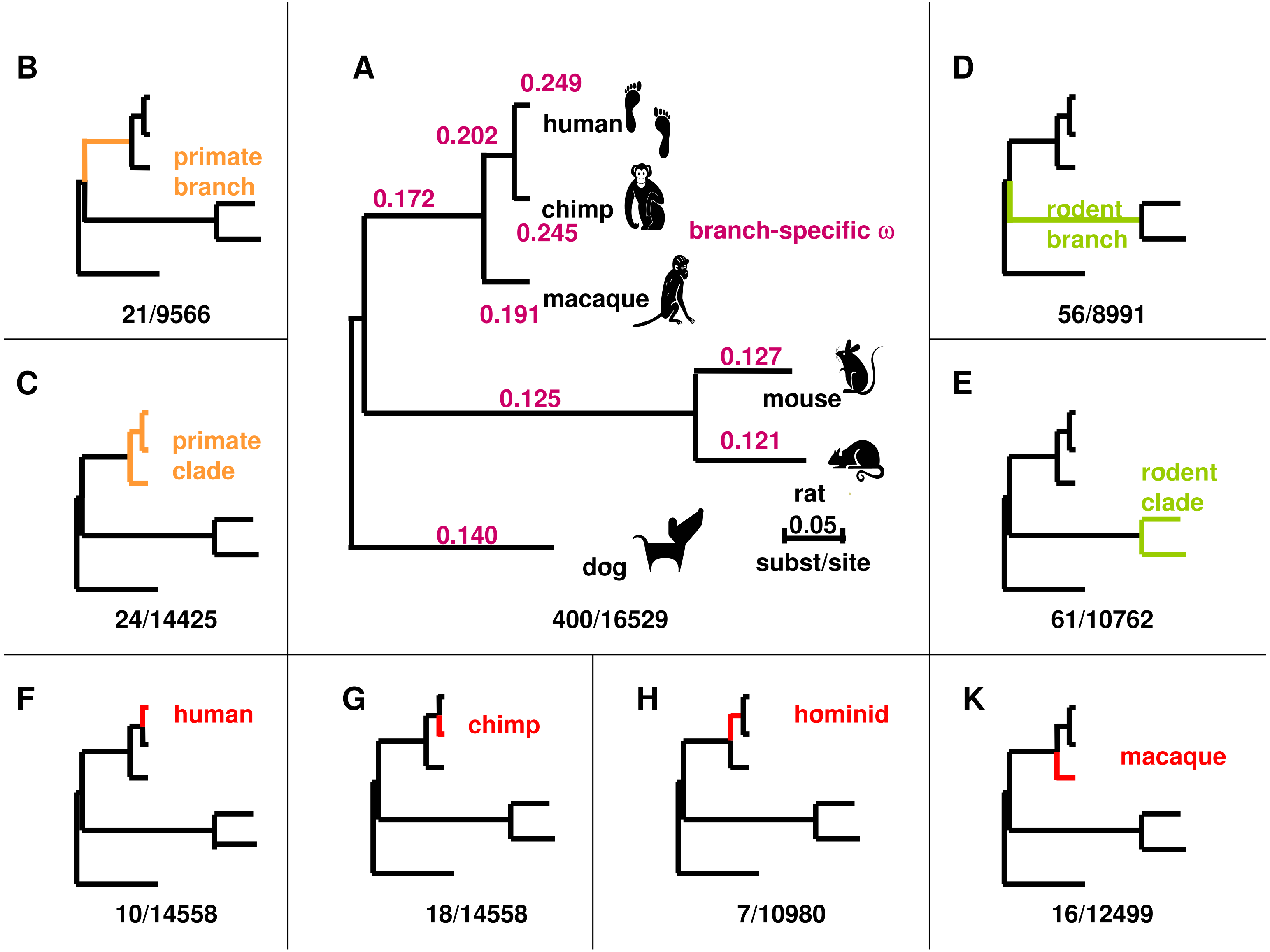
Ancestral sequences are not known:

$$\Pr(H,C,M\,|\,\omega,t_H,t_C,t_M) =$$
$$\sum_A \Pr(A,H,C,M\,|\,\omega,t_H,t_C,t_M)$$

Likelihood ω:

$$L(\omega) = \max_{t_H,t_C,t_M} \Pr(H,C,M\,|\,\omega,t_H,t_C,t_M)$$

- This likelihood can be computed e.g. by PAML software
- There are also more complex models,
  e.g. with ω varying within a gene


A
0.249
human
0.202
0.172
chimp
0.245
branch-specific ω
macaque
0.191
0.127
mouse
0.125
0.121
rat
0.05
subst/site
0.140
dog
400/16529
B
primate
branch
21/9566
C
primate
clade
24/14425
D
rodent
branch
56/8991
E
rodent
clade
61/10762
F
human
10/14558
G
chimp
18/14558
H
hominid
7/10980
K
macaque
16/12499

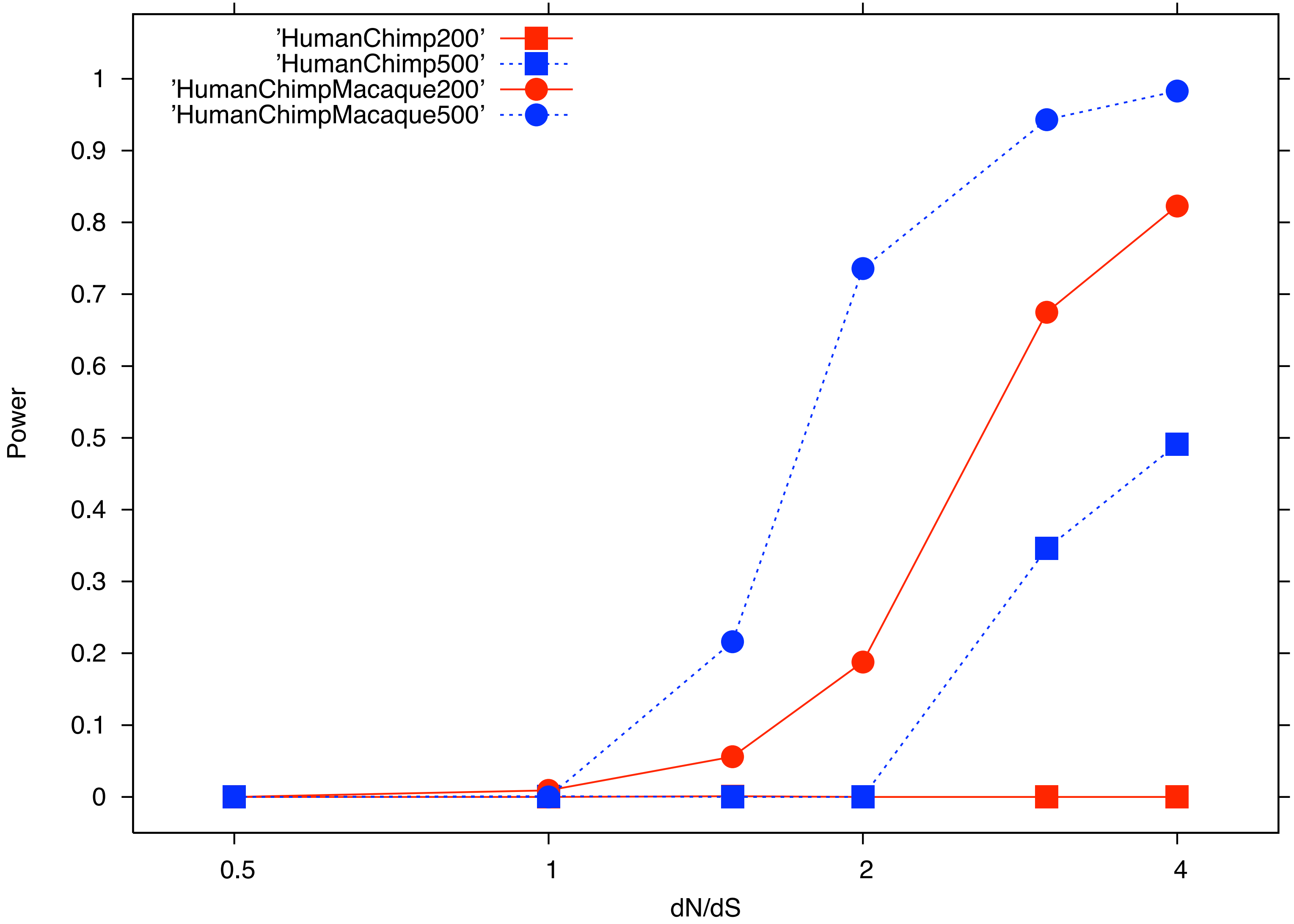
Functional categories enriched for positively selected genes

Defense: cellular defense response, antigen processing and presentation, response to virus, response to bacterium

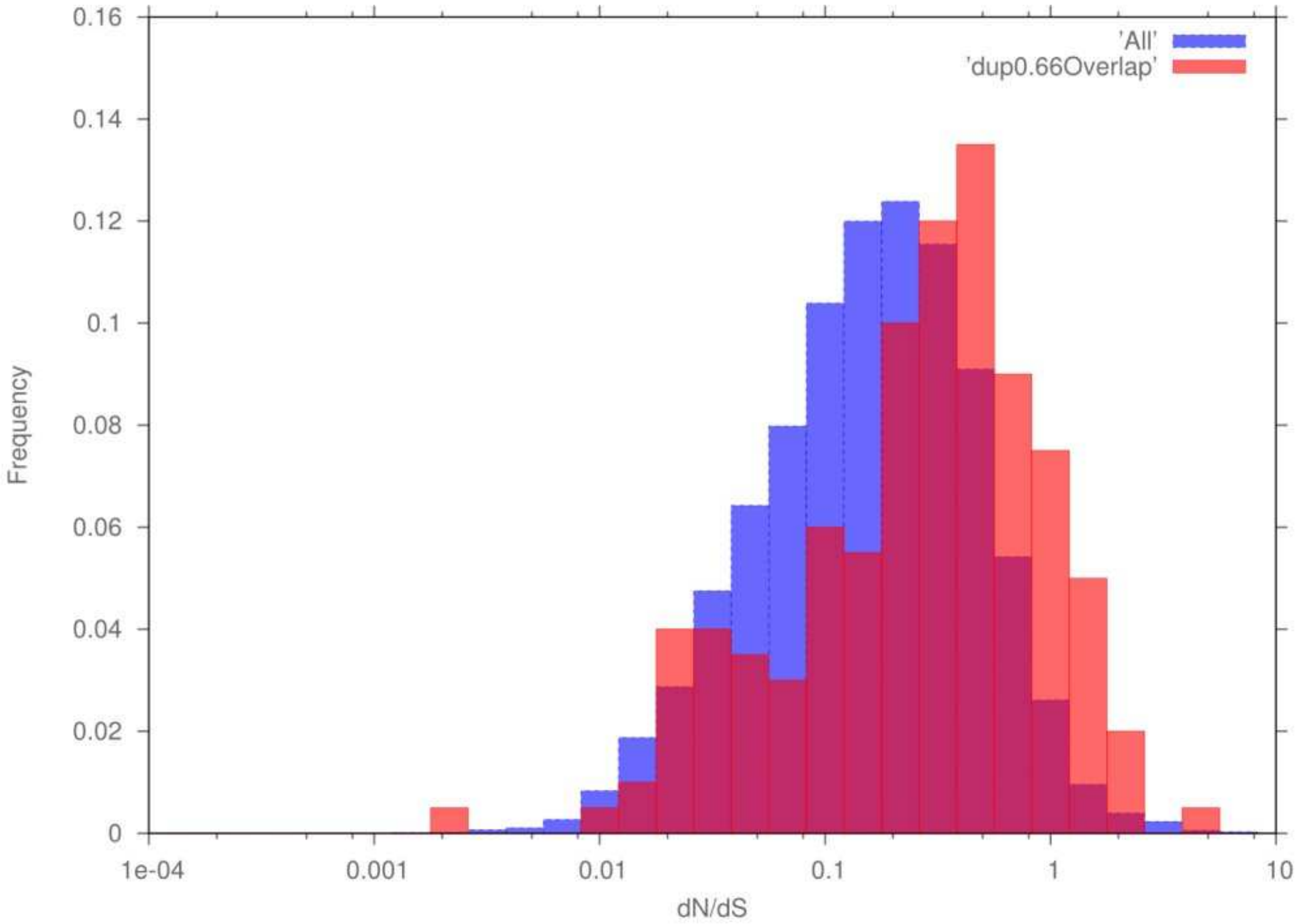
Immunity: adaptive immune response, adaptive immune response somatic recomb, lymphocyte mediated immunity, immunoglobulin mediated immune response, B cell mediated immunity, innate immune response, complement activation alternative pathway, regulation of immune system process, positive regulation of immune response, humoral immune response, complement activation classical pathway, humoral immune response circulating immunoglob, complement activation, activation of plasma proteins mute inflam resp, akute inflammatory response, response to wounding

Sensory perception: sensory perception of taste, G-protein coupled receptor protein signaling pathway, neurological process, sensory perception of chemical stimulus, sensory perception of smell

Adding genomes helps to improve power of the tests



Positive selection in duplicated genes



Summary

- Natural selection plays an important role in the evolution
- **Purifying selection:**
  - Conserved regions are likely to have some function
  - To find genes, we consider also typical codon mutations
- **Positive selection:**
  - Positive selection in genes causes high fraction of nonsynonymous changes (evolution at the protein level)
  - Duplicated genes are more often under positive selection
  - Hunt continues: we want to find genes causing human-specific features
- **Methods:** substitution models, phylogenetic HMMs, likelihood ratio tests

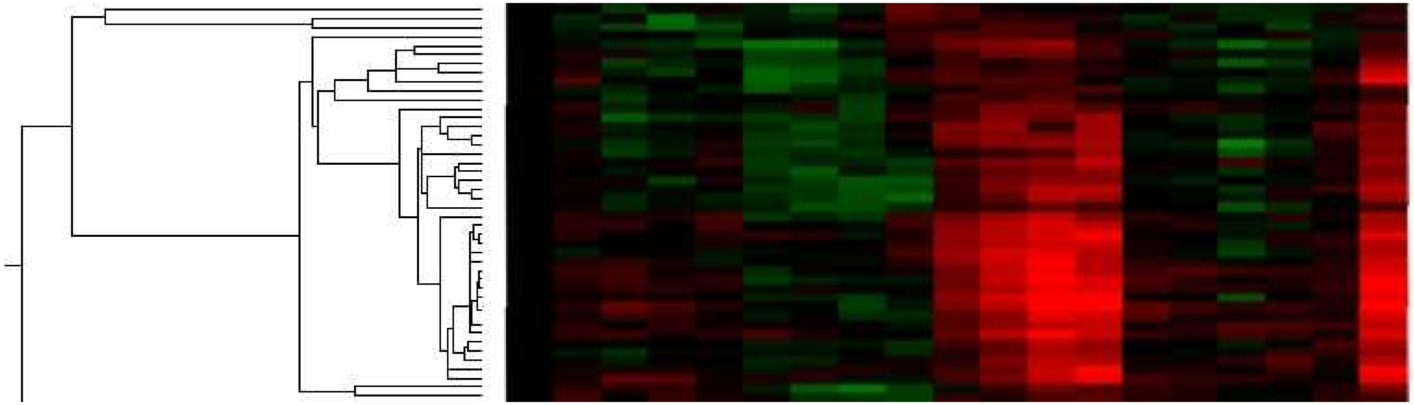
Announcements

- Homework 2 will be published today or tomorrow
- Homework 1 marks will eventually appear in Moodle
- Journal club meetings:
  group 4 done, group 2 agreed date,
  group 5 ??, group 6 looking for date

Regulation of Gene Expression

Broňa Brejová

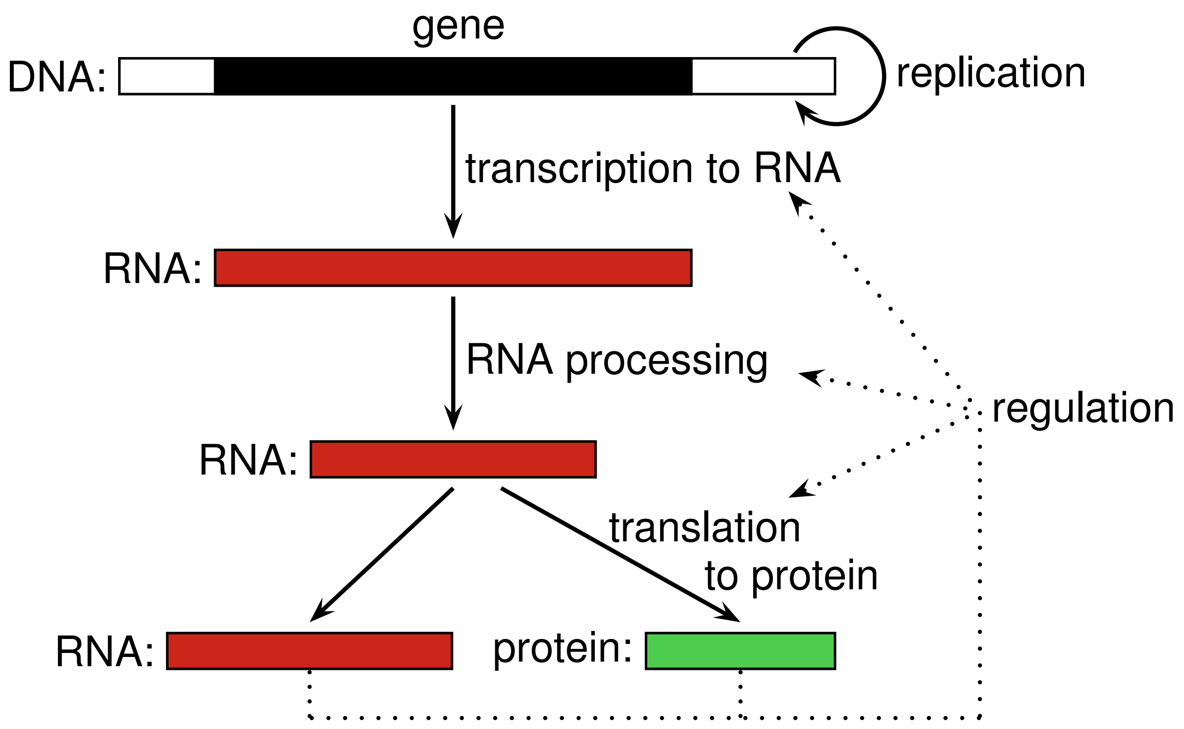
November 11, 2021



Recall: What information is stored in DNA?

Genes: Recipes for synthesis of proteins and functional RNAs.

Regulation of their expression: when and how much to synthesize

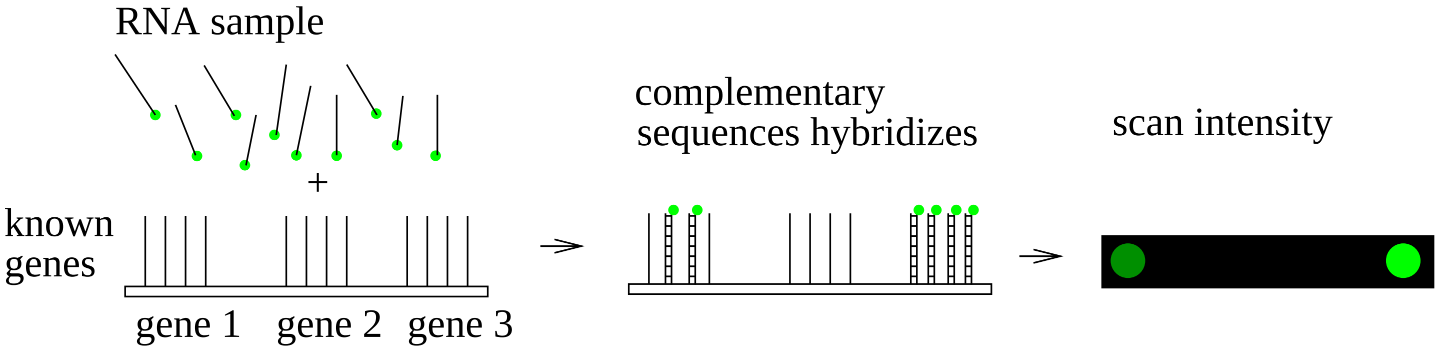


Regulation at the level of transcription, processing, translation, posttranslational modifications, . . .

Goals

- Determine under which conditions a gene is expressed (related to gene function)
- Which genes regulate it
- Details of the regulatory mechanism (binding sites, expression levels,. . .)

Technology: expression array, microarray



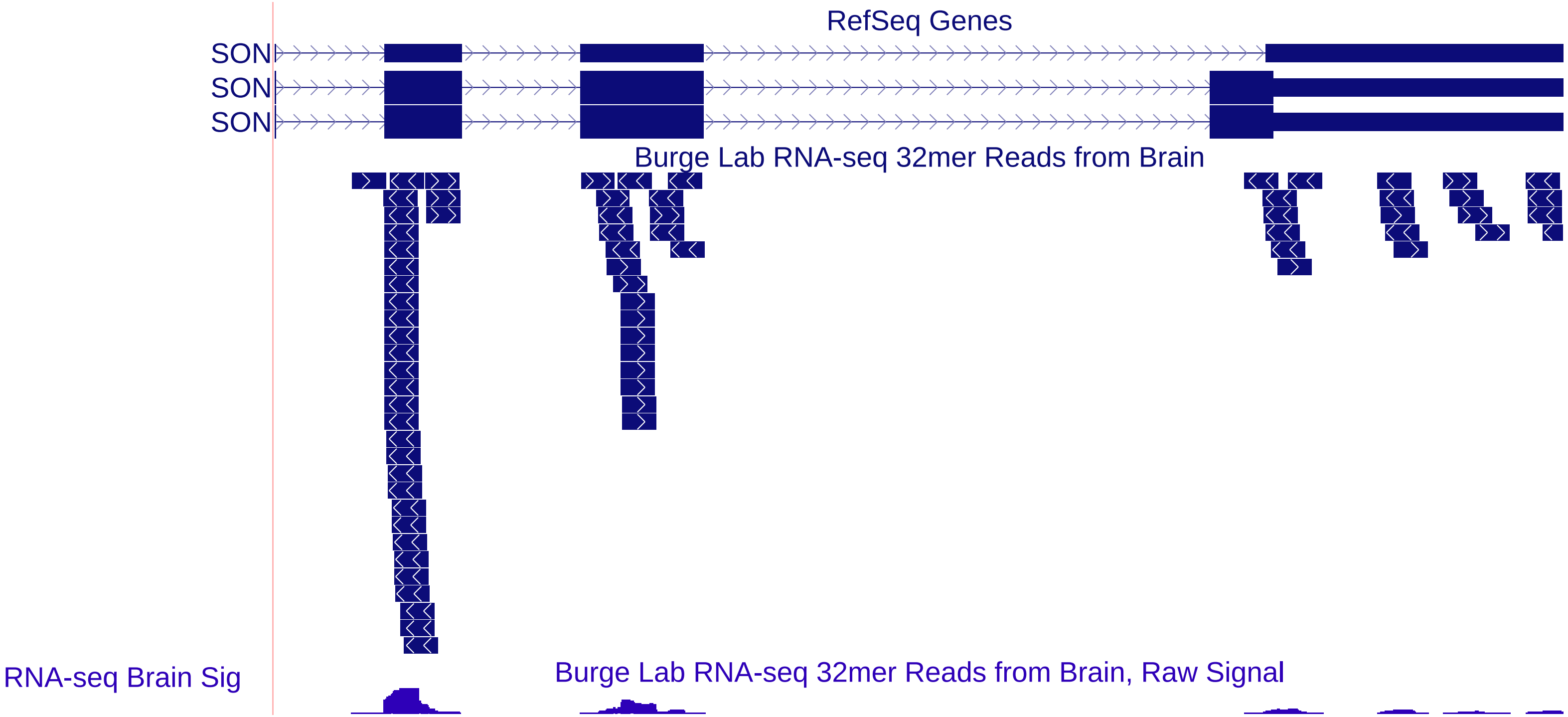
Measuring the amount of mRNA present in the sample for **many genes** at the same time.

Repeated under different conditions.

Technology: RNA-seq

Sequencing RNA extracted from the sample by NGS technologies, mapping reads to the genome.

The depth of coverage corresponds to the expression level


RefSeq Genes
SON
SON
SON
Burge Lab RNA-seq 32mer Reads from Brain
RNA-seq Brain Sig
Burge Lab RNA-seq 32mer Reads from Brain, Raw Signal

Example from the UCSC genome browser

Example of expression array data

Ratio of gene expression in sample and control fg/bg

	15min	30min	1h	2h	4h	...
W95909	0.72	0.1	0.57	1.08	0.66	
AA045003	1.58	1.05	1.15	1.22	0.54	
AA044605	1.1	0.97	1	0.9	0.67	
W88572	0.97	1	0.85	0.84	0.72	
AA029909	1.21	1.29	1.08	0.89	0.88	
AA059077	1.45	1.44	1.12	1.1	1.15	
...						

Iyer et al 1999 The Transcriptional Program in the Response of Human Fibroblasts to Serum

Fibroblasts: cells synthesizing components of extracellular matrix.

To divide, they need growth factors added as “fetal bovine serum”.

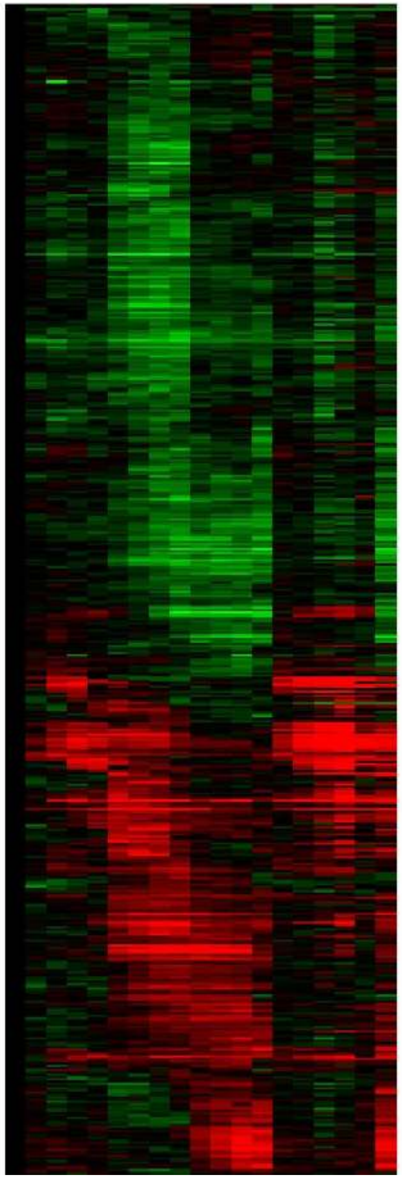
Visualization

Red: fg>bg

Green: fg<bg

517 genes (out of 8600)

19 experiments



This lecture: different type of data

Other lectures in this course: work with sequences

- genome assembly
- sequence alignment
- gene finding
- fylogenetic trees, population and comparative genomics
- structure and function of proteins and RNA

Today: table of numbers

- typical data in statistics
- we can use general methods of statistics and machine learning

The first set of problems: preprocessing data

- Read intensity from microarray images, detect invalid measurements
- Data aggregation from multiple measurements per gene
- Use of control probes
- Normalization to obtain data comparable across experiments

Microarray measurements are very noisy, many sources of errors

A simple result:

list of genes highly underexpressed/overexpressed
e.g. fg/bg> 2, or fg/bg< 0.5
often only these genes used for further analysis

Clustering (zhlukovanie)

Goal: find groups of genes with similar expression profiles.
If many genes in the group have the same function,
the remaining genes may participate as well

Measuring profile similarity: e.g. Pearson correlation coefficient
Profile of gene 1: $x_1, x_2, \ldots, x_n$, mean $\overline{x}$
Profile of gene 2: $y_1, y_2, \ldots, y_n$, mean $\overline{y}$

$$C(x,y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

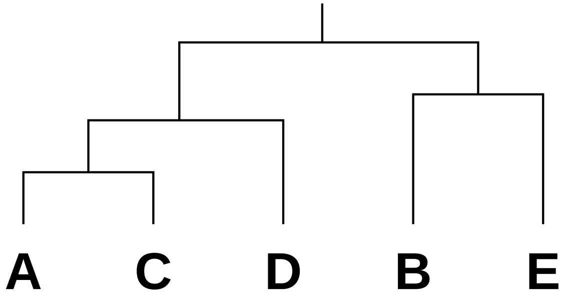
Number between -1 and 1, 1 for linearly correlated data
Distance $d(x,y) = 1 - C(x,y)$

Also other options, e.g. Euclidean distance

Hierarchical clustering

- Similar to neighbor joining method for building phylogenetic trees
- Start with each gene in a separate group
- Find two closest groups and join them to one
- Repeat until all genes are in one group
- Distance of two groups: e.g. distance of closest genes from one and the other group or average of distances over all pairs
- The result is a tree representing hierarchy of clusters

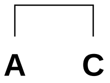
	A	B	C	D	E
gén A	0	0.6	0.1	0.3	0.7
gén B	0.6	0	0.5	0.5	0.4
gén C	0.1	0.5	0	0.6	0.6
gén D	0.3	0.5	0.6	0	0.8
gén E	0.7	0.4	0.6	0.8	0



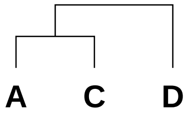
Hierarchical clustering - example

Distance of two groups: distance of closest genes from one and the other group (single linkage clustering)

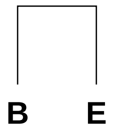
	A	B	C	D	E
gén A	0	0.6	0.1	0.3	0.7
gén B	0.6	0	0.5	0.5	0.4
gén C	0.1	0.5	0	0.6	0.6
gén D	0.3	0.5	0.6	0	0.8
gén E	0.7	0.4	0.6	0.8	0



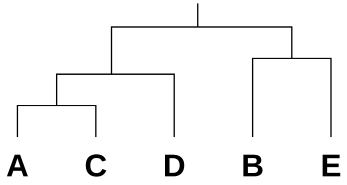
	A+C	B	D	E
A+C	0	0.5	0.3	0.6
B	0.5	0	0.5	0.4
D	0.3	0.5	0	0.8
E	0.6	0.4	0.8	0



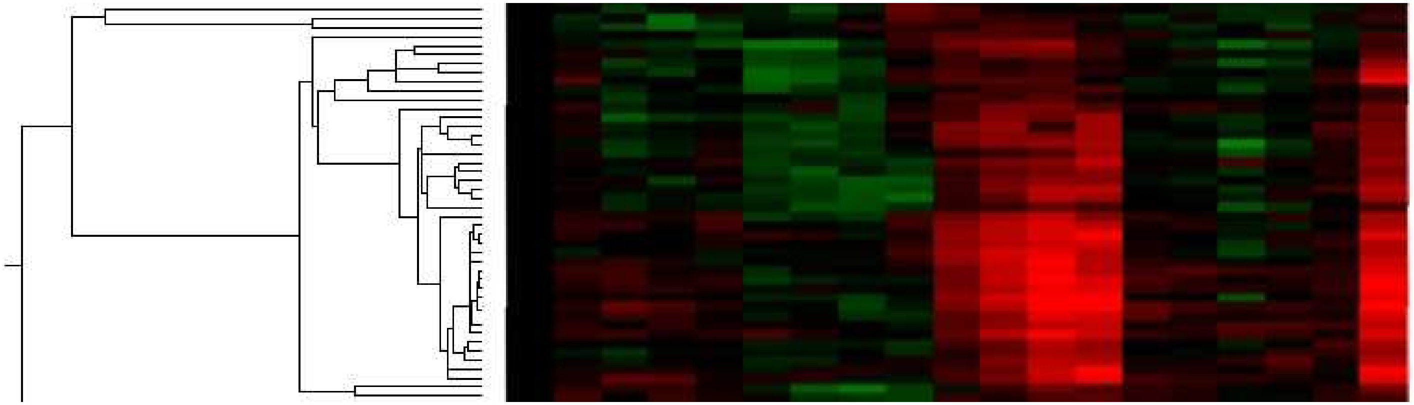
	A+C+D	B	E
A+C+D	0	0.5	0.6
B	0.5	0	0.4
E	0.6	0.4	0



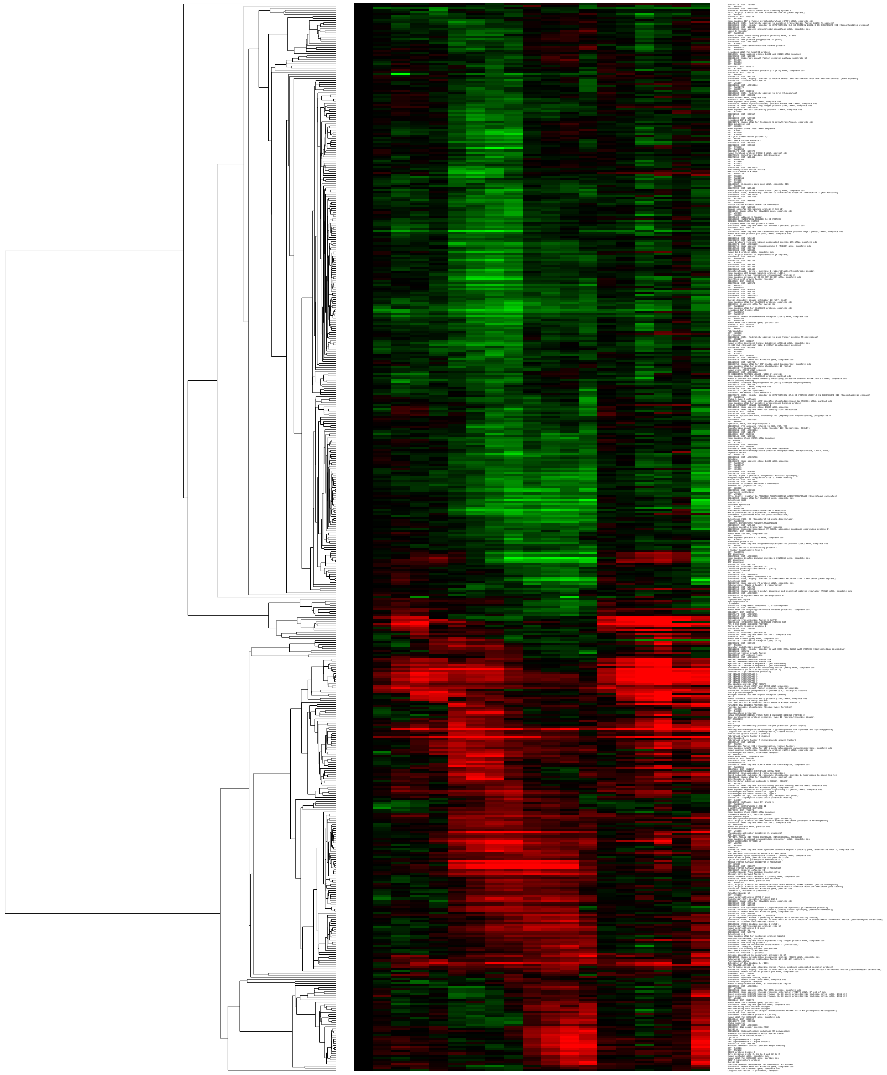
	A+C+D	B+E
A+C+D	0	0.5
B+E	0.5	0



Example: part of the microarray data



Clustering helps to visualize data,
similar genes get close to each other

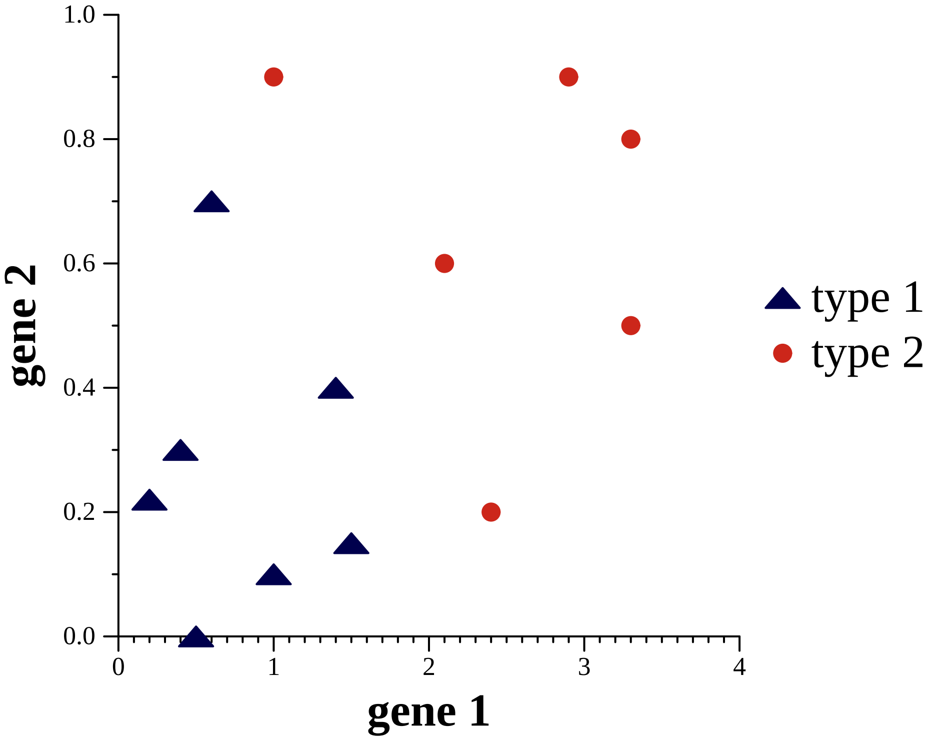


Classification

- A typical machine learning problem
- We might want to for example distinguish different types of tumors according to gene expression
- We are given examples with known expression and tumor type
- We want to find a formula which from the expression produces positive number for tumor type 1 and negative number for type 2
- We choose a family of functions with unknown parameters (hypothesis class)
- Find parameters that give the best accurracy on training data
- Accuracy of the resulting classifier tested on testing data (not used for training)
- The classifier then used on expression data with unknown tumor type

Toy example: expression of 2 genes

Training data with a known type:

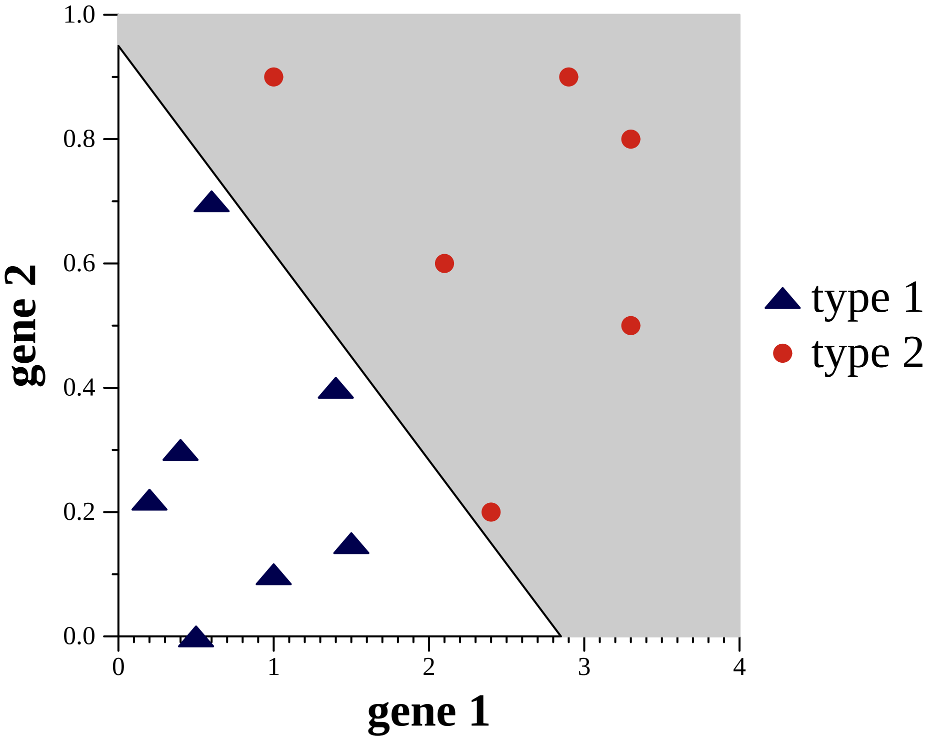


Hypothesis class: linear functions (linear discriminant)
type 1 tumor if $ax + by + c < 0$

The goal is to find a, b, c that work well on training data

Toy example: expression of 2 genes

Resulting classifier:



$a = 1, b = 3, c = -2.85$

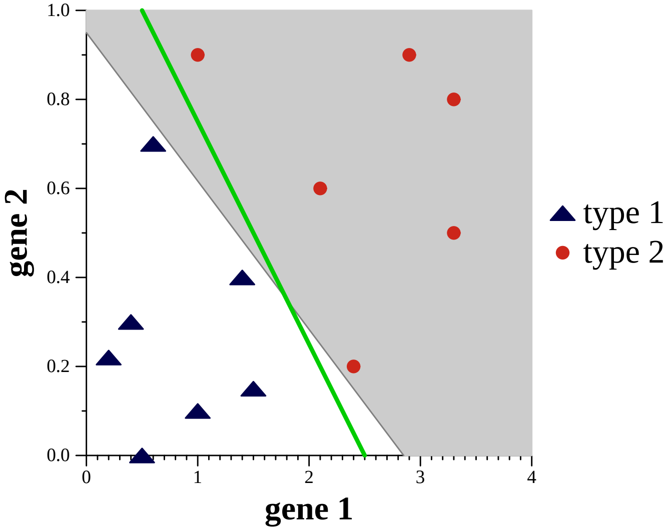
type 1 tumor if $x + 3y - 2.85 < 0$

Popular classification techniques

Logistic regression:

linear discriminant, assigns probability to each class, well-known method from statistics

Support vector machines (SVM): find linear discriminant with no training error which is most distant from all training examples



Can be generalized to non-linear functions by mapping vectors to a higher-dimensional space

Popular classification techniques

Neural networks:

“neurons” connected by “synapses”,
output of each neuron is a weighted combination of its inputs

Bayesian networks:

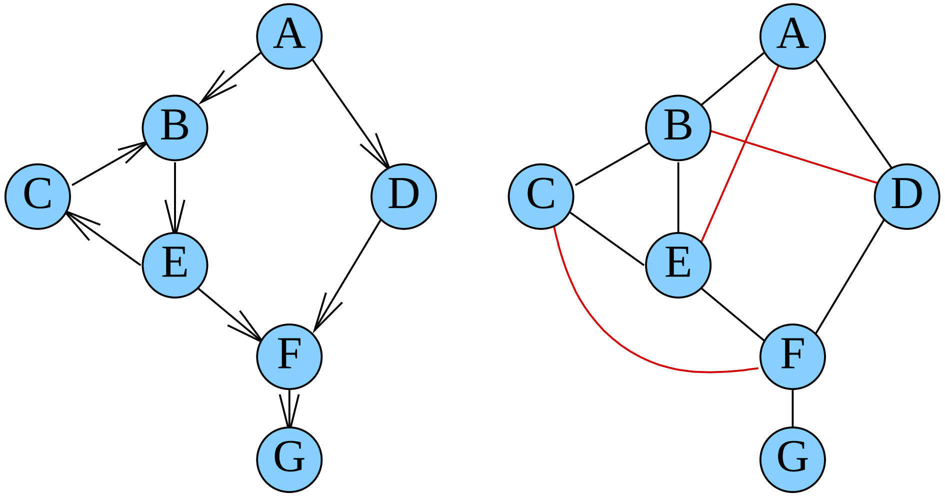
probabilistic model generating random expression profiles
tumor type also a random variable in the model with unknown state
similarly to a state in an HMM

Gene regulation network from expression data

Input: Expression profile for each gene, perhaps under known conditions (time series, deletion mutants)

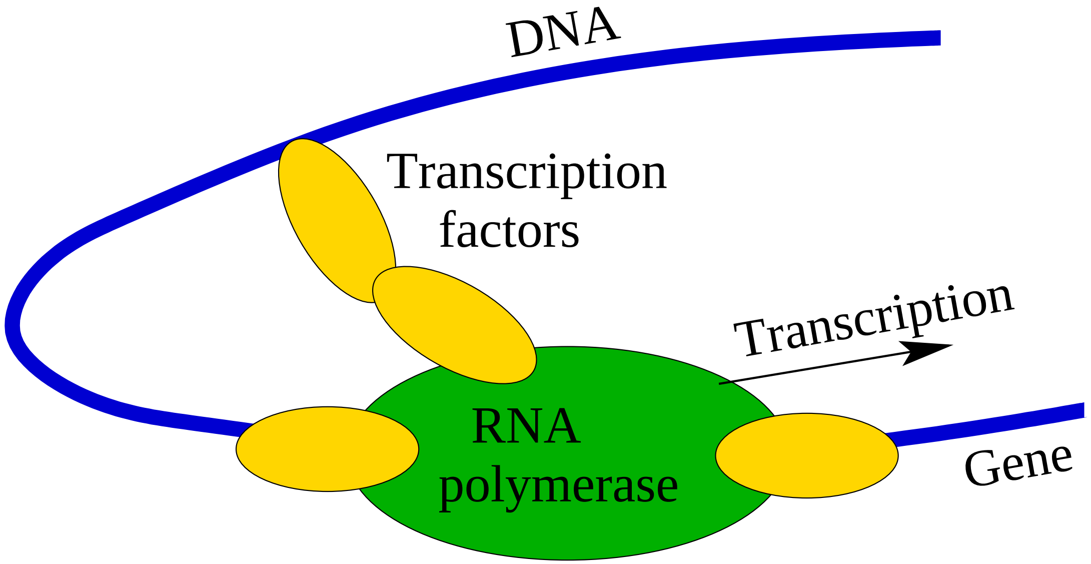
Output: Regulation network; nodes are genes, directed edge $A \to B$ if A regulates B

Expression profile similarity may provide undirected edges
The goal is to remove edges resulting from transitivity
and to direct edges correctly (difficult)



Transcription factors (TFs)

Regulation of transcription initiation by transcription factors: DNA binding proteins which help to attract RNA polymerase



Human genome has over 2000 TFs.

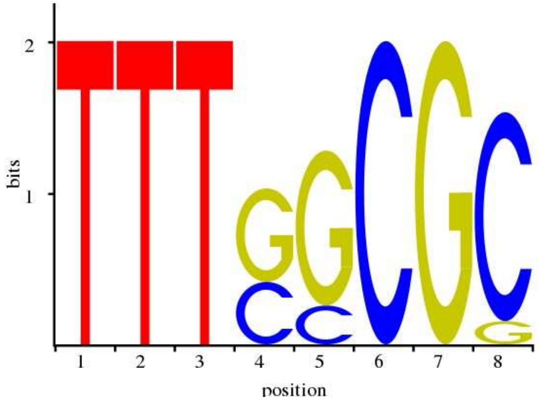
They can increase or decrese expression.

They can work in groups.

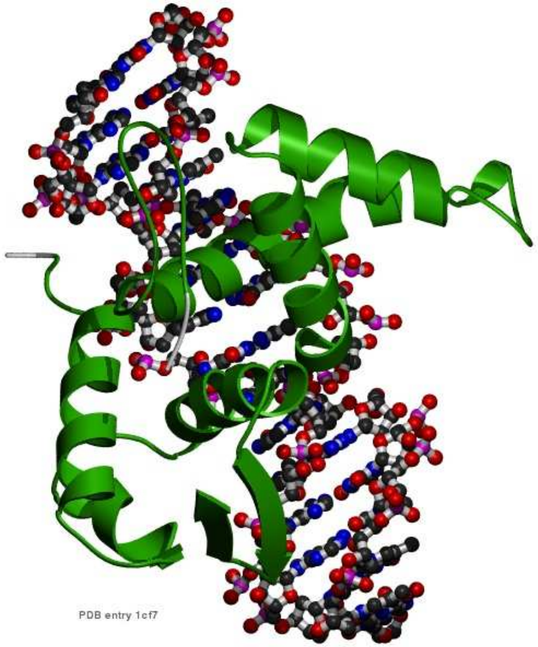
Example: E2F1 transcription factor

- Regulates cell cycle
- Binds TTTCCCGC, TTTCGCGC, and similar variants

A	0	0	0	0	0	0	0	0
C	0	0	0	4	2	10	0	9
G	0	0	0	6	8	0	10	1
T	10	10	10	0	0	0	0	0



- Goal: **represent** DNA sequences bound by a certain TF as a sequence **motif**, then search for **additional occurrences** in the genome


PDB entry 1cf7

Representation of binding motifs

String with mismatches (consensus):
motif is a string, occurrences can have a certain number of mismatches given in advance

Example: motif TTTGGCGC + 1 mismatch
TTTGGCGC, TT**A**GGCGC, TTTG**C**CGC are motif occurrences
TTT**CC**CGC not an occurrence

Choosing motif: take the most frequent letter at each position

A	0	0	0	0	0	0	0	0
C	0	0	0	4	2	10	0	9
G	0	0	0	6	8	0	10	1
T	10	10	10	0	0	0	0	0

Representation of binding motifs 2

Regular expression:

some positions specify character sets

[GC] means position where C or G is allowed

N means any base

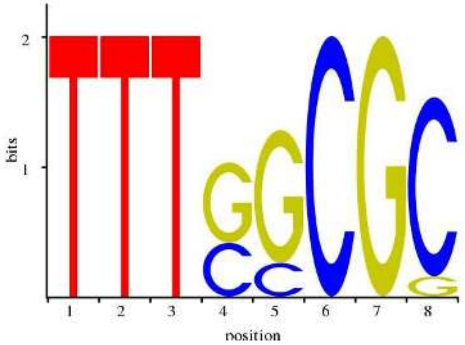
Example: motif TTT[CG][CG]CGC

TTTGGCGC, TTT**CC**CGC, TTTG**C**CGC are motif occurrences

TT**A**GGCGC is not an occurrence

Choosing motif: allow several most frequent letters at each position

A	0	0	0	0	0	0	0	0
C	0	0	0	4	2	10	0	9
G	0	0	0	6	8	0	10	1
T	10	10	10	0	0	0	0	0



Representation of binding motifs 3

Position specific scoring matrix (PSSM, PWM):

scoring matrix, score for each letter at each position

occurrences achieve score higher than threshold T

Example: $T = 8$

A	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0
C	-1.6	-1.6	-1.6	0.6	0.0	1.5	-1.6	1.4
G	-1.6	-1.6	-1.6	1.0	1.3	-1.6	1.5	-0.5
T	1.1	1.1	1.1	-2.0	-2.0	-2.0	-2.0	-2.0

TTT**CC**CGC is an occurrence: 1.1+1.1+1.1+0.6+0.0+1.5+1.5+1.4=8.3

TTTGGCG**G** is an occurrence: 1.1+1.1+1.1+1.0+1.3+1.5+1.5-0.5=8.1

TT**A**GGCGC is not: 1.1+1.1-2.0+1.0+1.3+1.5+1.5+1.4=6.4

Construction of PSSM: next lecture

Finding occurrences in the genome

- Consider motif in one of the representations:
  - Consensus, e.g. TTTGGCGC + 1 mismatch
  - Regular expression, e.g. TTT[CG][CG]CGC
  - Scoring matrix, e.g. threshold $T = 8$ and matrix:

A	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0	-2.0
C	-1.6	-1.6	-1.6	0.6	0.0	1.5	-1.6	1.4
G	-1.6	-1.6	-1.6	1.0	1.3	-1.6	1.5	-0.5
T	1.1	1.1	1.1	-2.0	-2.0	-2.0	-2.0	-2.0

- Test each position in the genome if it is an occurrence
- Occurrences are potential binding sites

Finding occurrences in the genome: problem

- Test each position in the genome if it is a motif occurrence
- Besides **binding sites**, often also many **random occurrences**
- E-value of a motif: how many occurrences are expected in a random sequence
- For example TTT[CG][CG]CGC appears about once in 30,000 bases
- To improve specificity, we can search for
  - clusters of binding sites
  - sites validated by experiments
  - evolutionarily conserved sites
- Motif databases, e.g. TRANSFAC, JASPAR

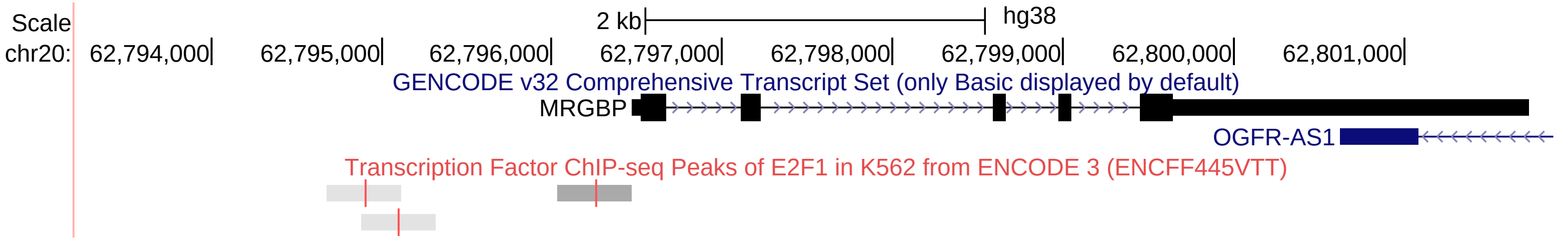
How to find binding sites experimentally?

Chromatin immunoprecipitation (ChIP)

Using an antibody specific to a given TF, we can determine approximate locations of its binding sites

- TF and DNA crosslinked by formaldehyde
- DNA cut to shorter segments
- Segments with crosslinked TF are bound by the antibody
- DNA is isolated and sequenced (**ChIP-seq**)

Problem: we find only approximate location of the binding site



Scale
2 kb
hg38
chr20:
62,794,000
62,795,000
62,796,000
62,797,000
62,798,000
62,799,000
62,800,000
62,801,000
GENCODE v32 Comprehensive Transcript Set (only Basic displayed by default)
MRGBP
OGFR-AS1
Transcription Factor ChIP-seq Peaks of E2F1 in K562 from ENCODE 3 (ENCFF445VTT)

How to find motifs by computational methods?

. . . without having several examples of a binding site

- Assume we have a group of sequences, each containing a binding site of the same TF, but binding preferences of this TF not known
- The goal is to find **the most specific** motif, occurring in all sequences or occurring more frequently than expected
- **Currently:** using ChIP-seq obtain regions of DNA surrounding binding sites, find motifs to refine the binding site position
- **Originally:** take a group of genes with similar expression profiles, thus possibly regulated by the same TF
  find motifs in DNA regions upstream of these genes

Consensus Pattern Problem (CPP)

Simple formulation of the motif finding problem

Input: motif length L, sequences $S_1, S_2, \ldots, S_k$

Output: motif (string) M of length L
and motif occurrence in each S_i (string s_i of length L)
such that the overall number of mismatches between M and s_i) is smallest possible

Example:
Input: CAAACAT, AGTAGC, TAACCA, TCTCCTC, $L = 4$
Output: motif TAAC
Occurrences and mismatches AAAC 1, TAGC 1, TAAC 0, TCTC 2
Total mismatches 4

Solving CPP

NP-hard problem

- **Idea 1:** Try all possible motifs of length L

  Problem: Not practical — why?

- **Idea 2:** Try all substrings of length L of input strings $S_1, \ldots, S_k$

  Problem: Sometimes gives wrong answer — why?

  But this always finds a solution
  with cost at most twice the optimum
  (2-approximation algorithm)

- **Further improvements:**
  Try consensus sequences
  of all samples of r substrings from input
  PTAS (polynomial-time approximation scheme)

Input: $L = 4$
CAAACAT,
AGTAGC,
TAACCA,
TCTCCTC

Output:
motif TAAC
Occurrences
and mismatches:
AAAC 1,
TAGC 1,
TAAC 0,
TCTC 2
Total mismatches 4

A more practical approach to motif finding

Probabilistc model generating sequence S using matrix W of base frequences in the motif and background frequences q outside the motif

A	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
C	0.01	0.01	0.01	0.39	0.19	0.97	0.01	0.01	0.89
G	0.01	0.01	0.01	0.59	0.79	0.01	0.97	0.97	0.09
T	0.97	0.97	0.97	0.01	0.01	0.01	0.01	0.01	0.01

$q(A) = 0.3$, $q(C) = 0.2$, $q(G) = 0.2$, $q(T) = 0.3$

Motif position in S is chosen randomly and each base is then generated according to q or one column of W

This model defines $\Pr(S \mid W)$.

Motif finding based on probabilistic models

Input: motif length L, sequences $S_1, \ldots, S_k$, frequencies q

Output: motif as a frequence matrix M maximizing likelihood $\Pr(S_1|W) \cdot \ldots \cdot \Pr(S_k|W)$

- Hard problem, addressed by heuristic algorithms
- For example EM (expectation maximalization)
- Local optimization, converging to a local maximum of likelihood
- Software: MEME

EM algorithm overview

- **Initialization:**
  Choose initial matrix W
  (e.g. based on one input substring of length L)

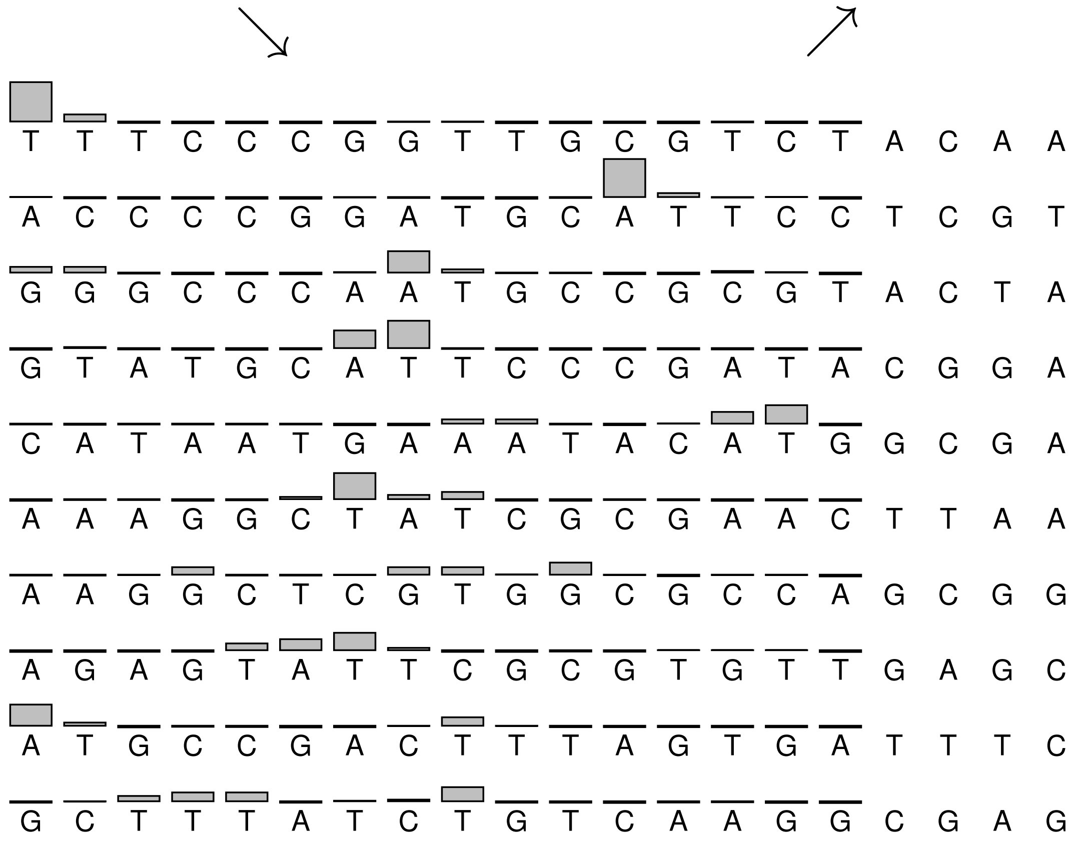
- **Iteration:**
  1. Assign each position j in sequence S_i weight $p_{i,j}$ corresponding to probability that $S_i[j]$ is a start of the motif W.
  2. Compute W from all possible occurrences in $S_1, \ldots, S_k$ weighted by $p_{i,j}$

Iterations increase likelihood until convergence.
Repeat, starting from many different starting values W

Example of the EM algorithm

A	0.10	0.10	0.10	0.10	0.10
C	0.10	0.10	0.10	0.70	0.70
G	0.10	0.10	0.10	0.10	0.10
T	0.70	0.70	0.70	0.10	0.10

A	0.31	0.14	0.06	0.07	0.07
C	0.06	0.10	0.19	0.71	0.61
G	0.12	0.17	0.29	0.14	0.25
T	0.51	0.60	0.46	0.08	0.07



T T T C C C G G T T G C G T C T A C A A

A C C C C G G A T G C A T T C C T C G T

G G G C C C A A T G C C G C G T A C T A

G T A T G C A T T C C C G A T A C G G A

C A T A A T G A A A T A C A T G G C G A

A A A G G C T A T C G C G A A C T T A A

A A G G C T C G T G G C G C C A G C G G

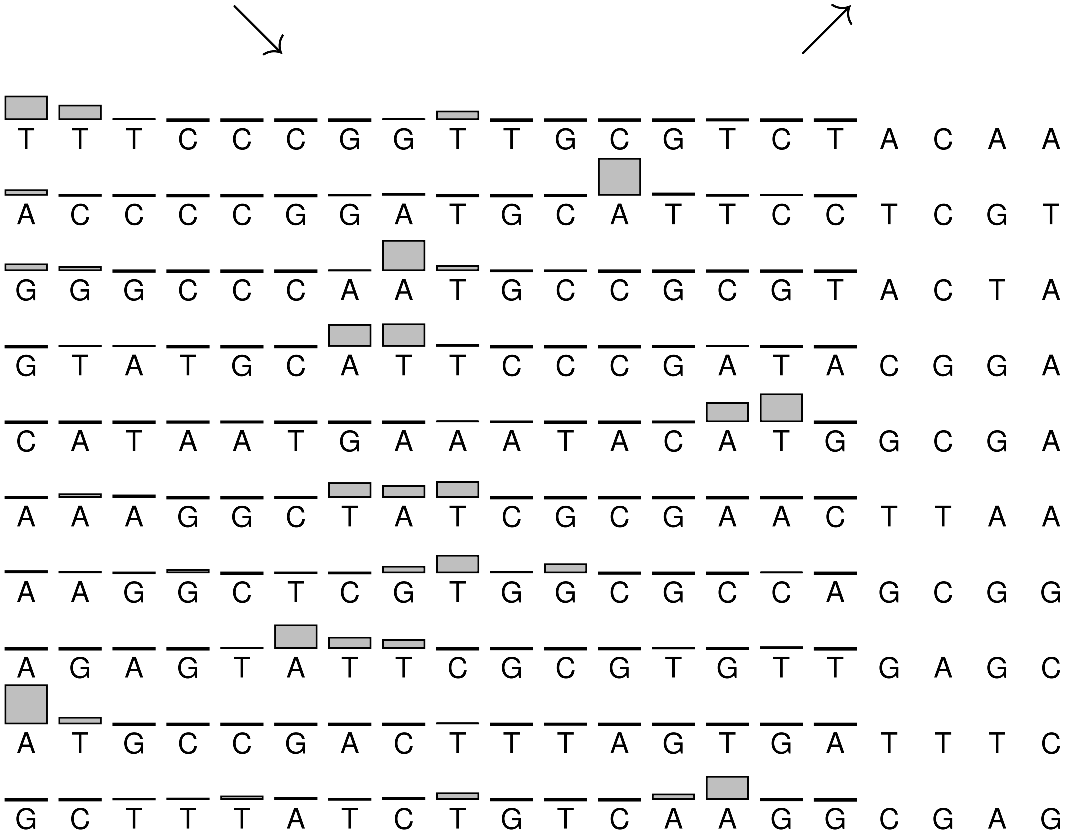
A G A G T A T T C G C G T G T T G A G C

A T G C C G A C T T T A G T G A T T T C

G C T T T A T C T G T C A A G G C G A G

Example of the EM algorithm: next iteration

A	0.31	0.14	0.06	0.07	0.07	A	0.47	0.09	0.01	0.02	0.03
C	0.06	0.10	0.19	0.71	0.61	C	0.02	0.11	0.20	0.80	0.58
G	0.12	0.17	0.29	0.14	0.25	G	0.08	0.22	0.48	0.15	0.35
T	0.51	0.60	0.46	0.08	0.07	T	0.42	0.58	0.30	0.03	0.03



T T T C C C G G T T G C G T C T A C A A

A C C C C G G A T G C A T T C C T C G T

G G G C C C A A T G C C G C G T A C T A

G T A T G C A T T C C C G A T A C G G A

C A T A A T G A A A T A C A T G G C G A

A A A G G C T A T C G C G A A C T T A A

A A G G C T C G T G G C G C C A G C G G

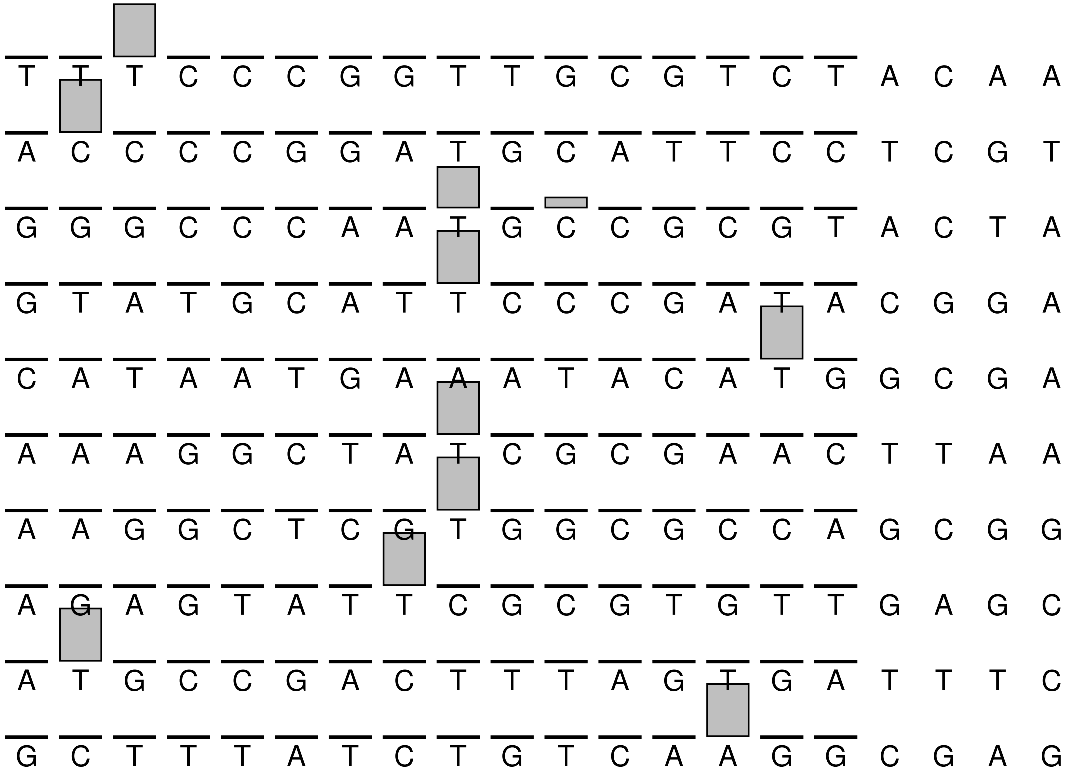
A G A G T A T T C G C G T G T T G A G C

A T G C C G A C T T T A G T G A T T T C

G C T T T A T C T G T C A A G G C G A G

Example of the EM algorithm: after 20 iterations

A	0.10	ϵ	ϵ	ϵ	ϵ
C	0.12	0.52	0.48	$1-3\epsilon$	ϵ
G	ϵ	0.48	0.52	ϵ	$1-3\epsilon$
T	0.78	ϵ	ϵ	ϵ	ϵ



T T T C C C G G T T G C G T C T A C A A

A C C C C G G A T G C A T T C C T C G T

G G G C C C A A T G C C G C G T A C T A

G T A T G C A T T C C C G A T A C G G A

C A T A A T G A A A T A C A T G G C G A

A A A G G C T A T C G C G A A C T T A A

A A G G C T C G T G G C G C C A G C G G

A G A G T A T T C G C G T G T T G A G C

A T G C C G A C T T T A G T G A T T T C

G C T T T A T C T G T C A A G G C G A G

Summary

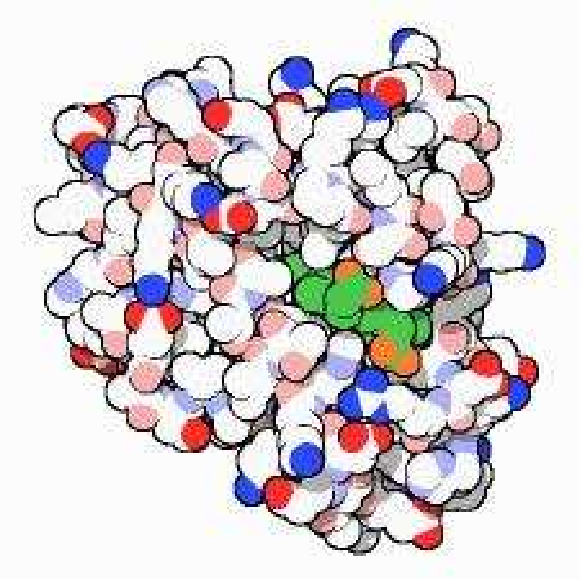
- Microarrays or RNA-seq can characterize expression levels of many genes at once, but produce noisy data
- Clustering (zhlukovanie) can find similar genes
  no prior training set is necessary (unsupervised learning)
- Classification can distinguish e.g. diseases according to expression
  needs training data with known answers (supervised learning)
- Expression data help to build regulatory networks
- Binding motifs can be represented in various forms
  (string, regular expression, scoring matrix)
- These motifs are not sufficiently specific, therefore it is hard to recognize binding sites in the genome
- EM algorithm for finding new motifs in sequences

Announcements

- Homework 2 published, submit until November 30 22:00
- Journal club meetings:
  group 4 done,
  groups 2,5 met, please write a short report
  group 6 meeting tonight

Protein structure and function

Broňa Brejová
November 18, 2021

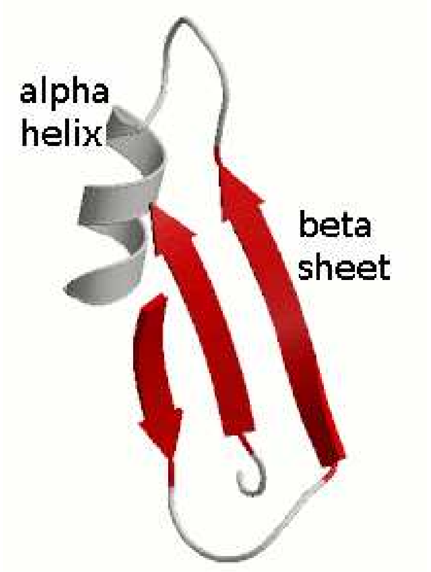


Proteins

Strings of 20 different amino acids with different chemical properties:

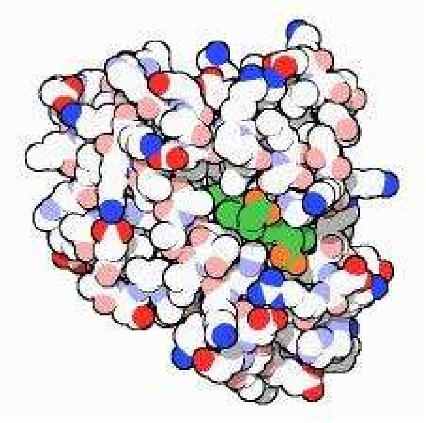
Amino Acid	Side chain	Its properties
Alanine (A)	-CH3	hydrophobic
Arginine (R)	-(CH2)3NH-C(NH)NH2	basic
Asparagine (N)	-CH2CONH2	hydrophilic
Aspartic acid (D)	-CH2COOH	acidic
Cysteine (C)	-CH2SH	hydrophobic
Glutamic acid (E)	-CH2CH2COOH	acidic
Glutamine (Q)	-CH2CH2CONH2	hydrophilic
Glycine (G)	-H	hydrophilic
Histidine (H)	-CH2-C3H3N2	basic
Isoleucine (I)	-CH(CH3)CH2CH3	hydrophobic
Leucine (L)	-CH2CH(CH3)2	hydrophobic
Lysine (K)	-(CH2)4NH2	basic
Methionine (M)	-CH2CH2SCH3	hydrophobic
Phenylalanine (F)	-CH2C6H5	hydrophobic
Proline (P)	-CH2CH2CH2-	hydrophobic
Serine (S)	-CH2OH	hydrophilic
Threonine (T)	-CH(OH)CH3	hydrophilic
Tryptophan (W)	-CH2C8H6N	hydrophobic
Tyrosine (Y)	-CH2-C6H4OH	hydrophobic
Valine (V)	-CH(CH3)2	hydrophobic

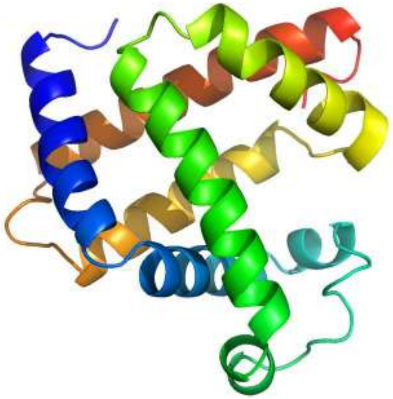
Protein structure



- **Primary structure:** sequence of amino acid
- **Secondary structure:** regular structural motifs
  alpha helix, beta sheet
- **Tertiary structure:** exact 3D positions of atoms
- **Quaternary structure:** interactions of several proteins in complex

Myoglobin, the first protein with a known structure
[Kendrew et al 1958]





Experimental structure determination

- X-ray crystallography
  – requires crystal form of the protein
- NMR (nuclear magnetic resonance spectroscopy)
  – mainly used on short proteins
- Cryo-EM (cryogenic electron microscopy)
  – less accurate, good for large protein complexes
- Expensive and difficult process
- Database of structures PDB
  184 000 protein structures
  (UniProt has over 200 million of sequences)

Bioinformatics problem: protein structure prediction, protein folding

Input: protein sequence

Output: 3D positions of atoms or amino acids

Ab initio methods

- Find a structure with the lowest free energy
- Physics-based formulas for approximating energy
  - forces among atoms of the protein and surrounding water
- Very hard computational problem
  - molecular dynamics simulation
  - optimization methods, e.g. gradient descent, simulated annealing
- Useful for short proteins and improving approximate structures

Practical approaches to protein structure prediction

For a **query protein**:

- Check if it has a **known structure** in PDB
- If not, try to find a **similar protein** in PDB (BLAST), query likely a similar structure
- If no appropriate BLAST match, try to find similar proteins by more sensitive approaches, **protein profiles** (this lecture)
- Even more distant homology can be found by **protein threading**
- Recently, approaches based on **deep learning** (neural networks) quite successful
- We can try to improve found structures by **energy minimization**
- **Predicted structures** can be also found in databases

Protein threading

- Even proteins with very different sequences can have similar structures
- We can try to “thread” the query protein to each known structure
- A special form of alignment taking into account interactions of amino acids in the known structure
- Computationally hard problem

Newest approaches: deep neural networks

- CASP competition every two years
- In 2018, 2020 won by AlphaFold designed by DeepMind/Google.
  In 2020, AlphaFold won by a large margin,
  predicted very well 2/3 of structures.
  It combines new ideas and existing approaches.
- Key idea used already before AlphaFold: **co-evolution detection**
  Find many homologs of the query protein
  (even if no structure known),
  build a multiple alignment,
  find positions that change together in evolution,
  these are potential 3D contacts

Newest approaches: deep neural networks

- **AlphaFold 1 (2018):**
  (1) Prediction of amino acid distances by a neural network.
  (2) Finding structure agreeing well with distances
  and an energy model using standard numerical optimization
  (gradient method) [animation]

- **AlphaFold 2 (2020):**
  combines both steps to a single neural network,
  which is run repeatedly on its outputs

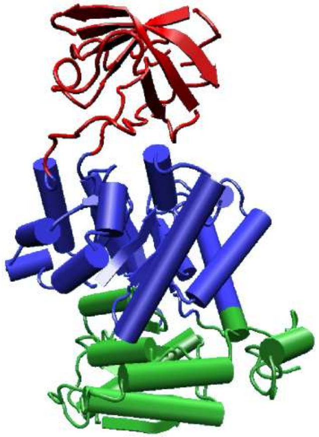
Recall: Practical approaches to protein structure prediction

For a **query protein**:

- Check if it has a **known structure** in PDB
- If not, try to find a **similar protein** in PDB (BLAST), query likely a similar structure
- If no appropriate BLAST match, try to find similar proteins by more sensitive approaches, **protein profiles** (this lecture)
- Even more distant homology can be found by **protein threading**
- Recently, approaches based on **deep learning** (neural networks) quite successful
- We can try to improve found structures by **energy minimization**
- **Predicted structures** can be also found in databases

Protein domains and families

Domain (doména)



- Part of a protein with an independent structure
- Many proteins contain multiple domains
- Domains can be rearranged during evolution

Family (rodina)

- Group of proteins or domains with similar sequence, structure and function
- If we know the structure of one family member, others might have a similar structure

Proteins as mosaics of domains

Pfam database

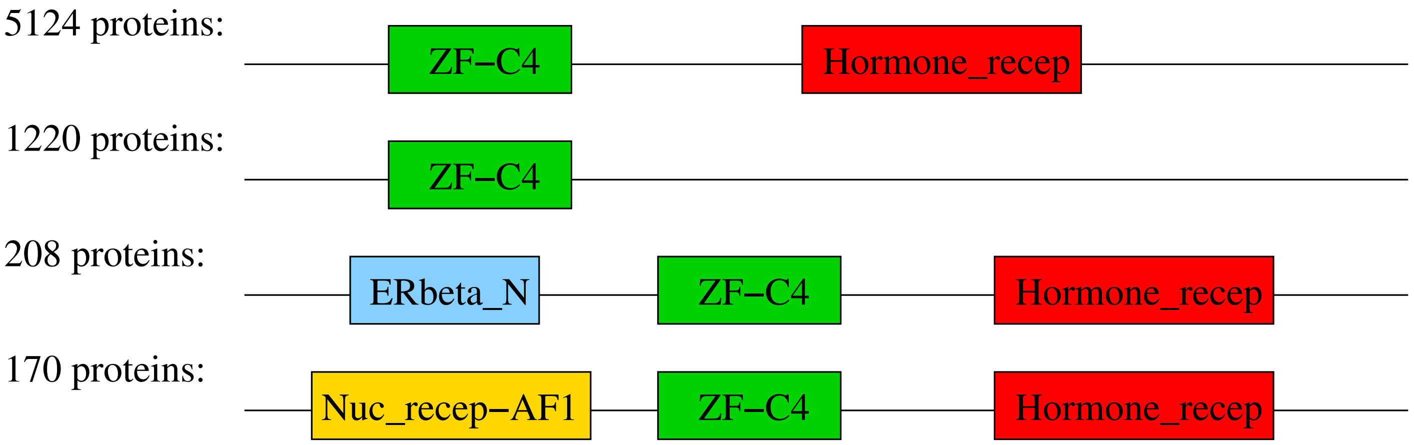
Domains in proteins classified to over 18 thousand families

77% of proteins have at least one known domain

53% protein sequences are covered by known domains

Example:

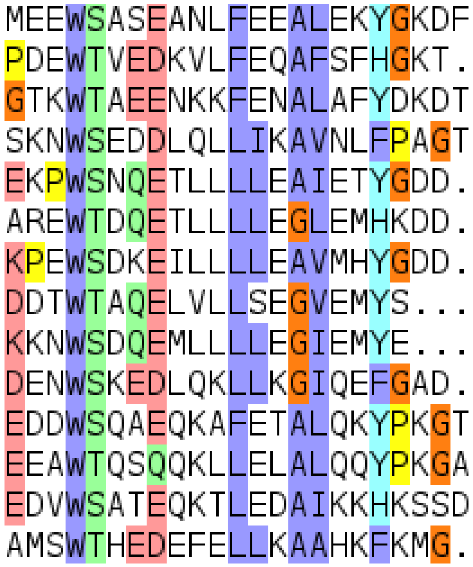
4 out of 91 architectures with Zinc finger, C4 type domain (Pfam)



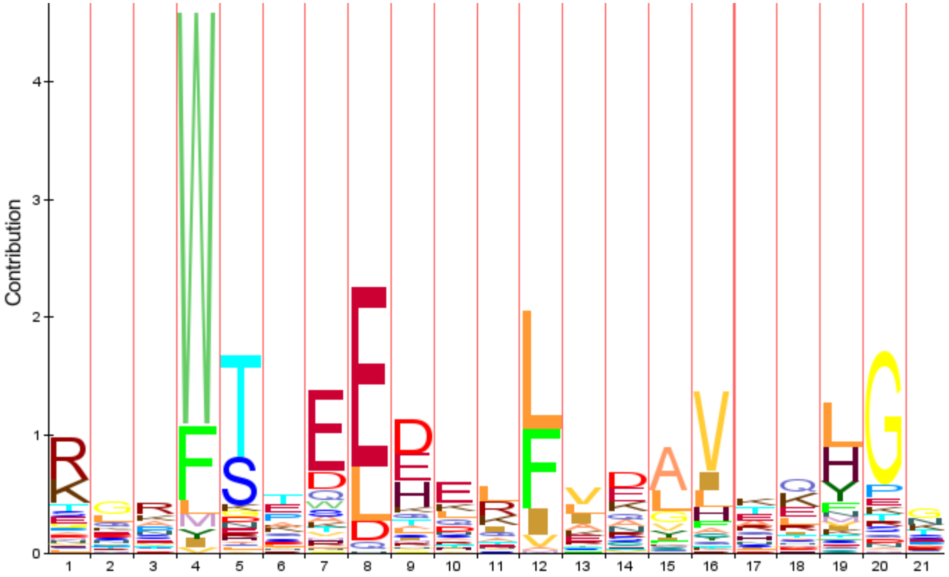
Characterization of a protein family

- Pairwise alignments (BLAST) between a query protein and family members do not always find weaker similarity

- Multiple sequence alignment of a family highlights important conserved positions



```
MEEWSASEANLFEEALEKYGKDF
PDEWTVEDKVLFEQAFSFHGKT.
GTKWTAEENKKFENALAFYDKDT
SKNWSEDDLQLLIKAVNLFPAGT
EKPWSNQETLLLLEAIETYGDD.
AREWTDQETLLLLEGLEMHKDD.
KPEWSDKEILLLLEAVMHYGDD.
DDTWTAQELVLLSEGVEMYS...
KKNWSDQEMLLLLEGIEMYE...
DENWSKEDLQKLLKGIQEFGAD.
EDDWSQAEQKAFETALQKYPKGT
EEAWTQSQQKLLELALQQYPKGA
EDVWSATEQKTLEDAIKKHKSSD
AMSWTHEDEFELLKAAHKFKMG.
```



Probabilistic profile of a family

(profile, position specific score matrix PSSM)

- In an alignment, compute $e_i(x)$: frequency of amino acid x in column i
- Create a model which generates sequence $x_1, x_2, \ldots, x_n$ with probability
$$e_1(x_1) \cdot e_2(x_2) \cdots e_n(x_n)$$
- Background model: sequence was generated randomly with amino acid x having frequency $q(x)$
- Score: log likelihood ratio in the two models

$$\log \frac{\prod_{i=1}^{n} e_i(x_i)}{\prod_{i=1}^{n} q(x_i)} = \sum_{i=1}^{n} \log \frac{e_i(x_i)}{q(x_i)} = \sum_{i=1}^{n} s_i(x_i)$$

Toy example of an PSSM

- Consider only leucine L a alanine A
- Multiple alignment of 10 sequences has the following counts:

	1	2	3	4
A	2	6	9	1
L	8	4	1	9

- Background model $q(A) = 30\%$, $q(L) = 70\%$
- Probability of sequence LAAL
  - in the profile model: $0.8 \cdot 0.6 \cdot 0.9 \cdot 0.9 = 0.3888$,
  - in the background model: $0.7 \cdot 0.3 \cdot 0.3 \cdot 0.7 = 0.0441$
- Score for LAAL: $\log_2(0.3888/0.0441) = 3.14$
- Score for LALA: $\log_2(0.0048/0.0441) = -3.20$

Toy example of an PSSM

- Multiple alignment of 10 sequences has the following counts:

	1	2	3	4
A	2	6	9	1
L	8	4	1	9

- Background model $q(A) = 30\%$, $q(L) = 70\%$
- Score of alanine in column 1: $s_1(A) = \log_2(0.2/0.3) = -0.58$,
  score of leucine in column 1: $s_1(L) = \log_2(0.8/0.7) = 0.19$
- Entire score table:

	1	2	3	4
A	-0.58	1.00	1.58	-1.58
L	0.19	-0.81	-2.81	0.36

- Score of LAAL is $0.19 + 1 + 1.58 + 0.36 = 3.13$
  Score of LALA is $0.19 + 1 - 2.81 - 1.58 = -3.20$

Pseudocounts

If some amino acid is completely absent at a given position, it would get probability 0 in the model

	1	2	3	4
A	2	6	9	0
L	8	4	1	10

To avoid this problem, add a small value, pseudocunt, to each count in the table (e.g. add 0.5):

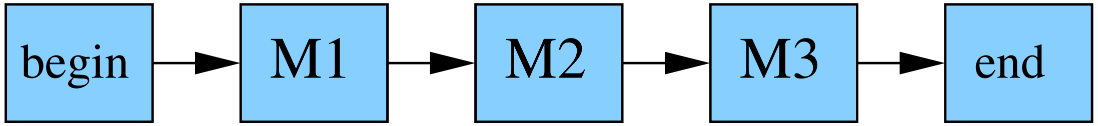
	1	2	3	4
A	2.5	6.5	9.5	0.5
L	8.5	4.5	1.5	10.5

Then compute scores as before

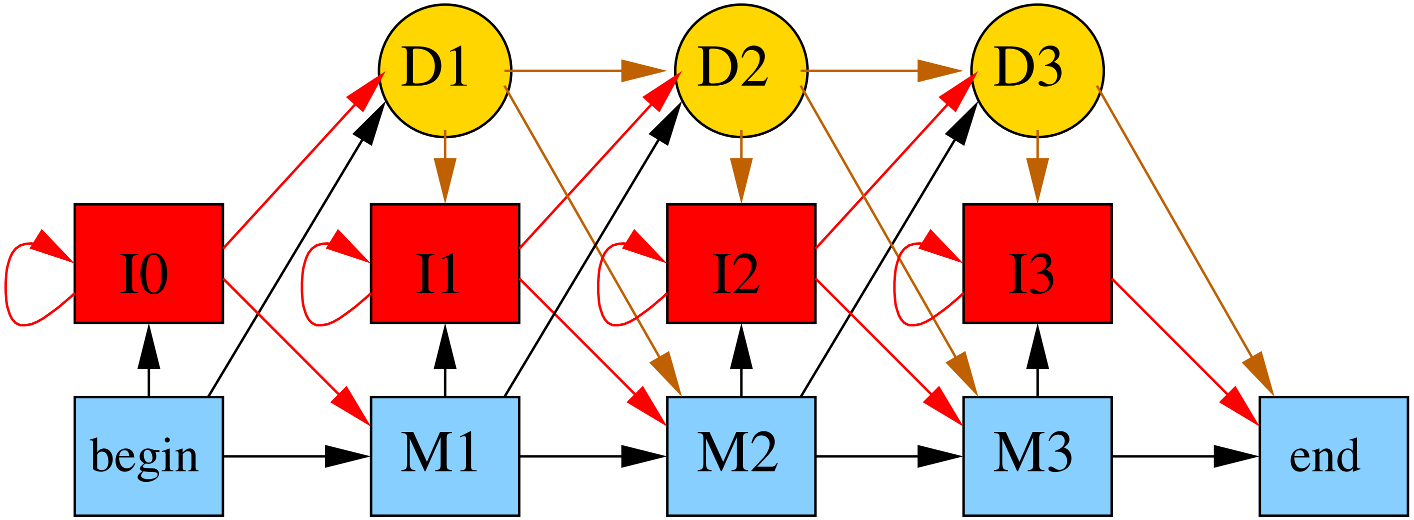
Profile HMMs (profilové HMM)

Extend profiles with insertions and deletions

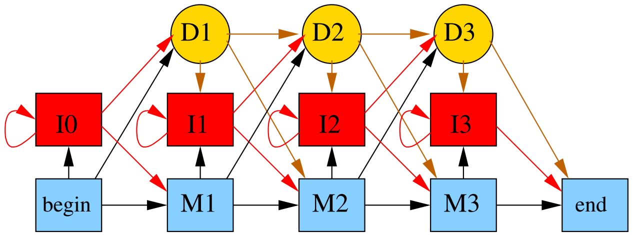
PSSM as an HMM:



Profile HMM: match state, insert state, delete state



Constructing profile HMMs



- Start from a multiple alignment
- Columns with a small fraction of gaps converted to match states, remaining columns handled by insert states
- In each column compute $E_i(a)$: the number of occurrences of a
- Emission probability $e_i(a) = \frac{E_i(a)}{\sum_b E_i(b)}$
- We add pseudocounts to avoid zero probabilities, $e_i(a) = \frac{E_i(a)+c}{\sum_b (E_i(b)+c)}$
- Transition probabilities set according to gaps
- Groups of very similar sequences used with lower weights

Using profiles and profile HMMs

Where to get profiles / profile HMMs?

- Pfam database contains domain families represented as profile HMMs
- PSI-Blast creates PSSMs on the fly from similar proteins
- PSSMs are also used to present binding site motifs in DNA (lecture on regulation)

How to find profile occurrences in a protein sequence?

- Similar to local alignemnt
- PSSM profiles: dynamic programming with fixed gap scores
- Profile HMMs: Viterbi/forward algorithms

Use the resulting score / probability to decide if a protein belongs to the family

Recall: Practical approaches to protein structure prediction

For a **query protein**:

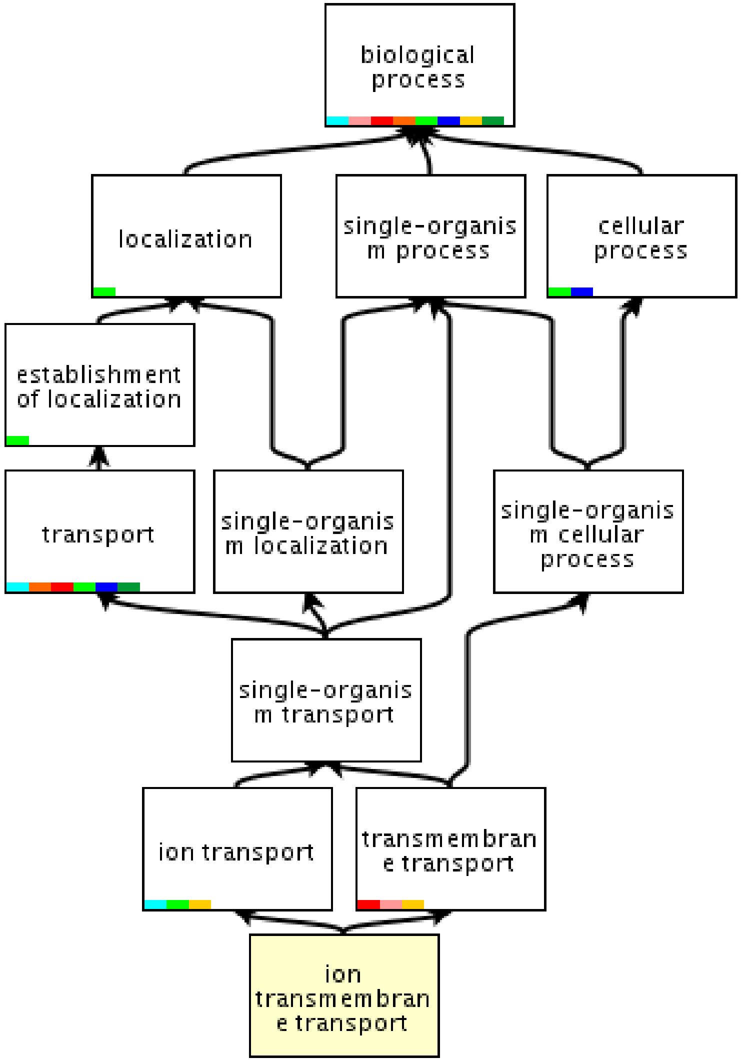
- Check if it has a **known structure** in PDB
- If not, try to find a **similar protein** in PDB (BLAST), our query likely has a similar structure
- If no appropriate BLAST match, try to find similar proteins by more sensitive approaches, **protein profiles** (this lecture)
- Even more distant homology can be found by **protein threading**
- Recently, approaches based on **deep learning** (neural networks) quite successful
- We can try to improve found structures by **energy minimization**
- **Predicted structures** can be also found in databases

Protein function

- Determined experimentally for some proteins
- Transfered to other proteins based on sequence similarity, domains, position in the genome and other data
- Swissprot/Uniprot collects known information about protein function
- Protein classification using Gene ontology (GO)
  Example of a term in GO:
  Accession: GO:0034220
  Name: ion transmembrane transport
  Ontology: biological_process
  Definition: A process in which an ion is transported from one side of a membrane to the other by means of some agent such as a transporter or pore.
  Comment: Note that this term is not intended for use in annotating lateral movement within membranes.

Gene ontology (GO)

Hierarchy of terms:

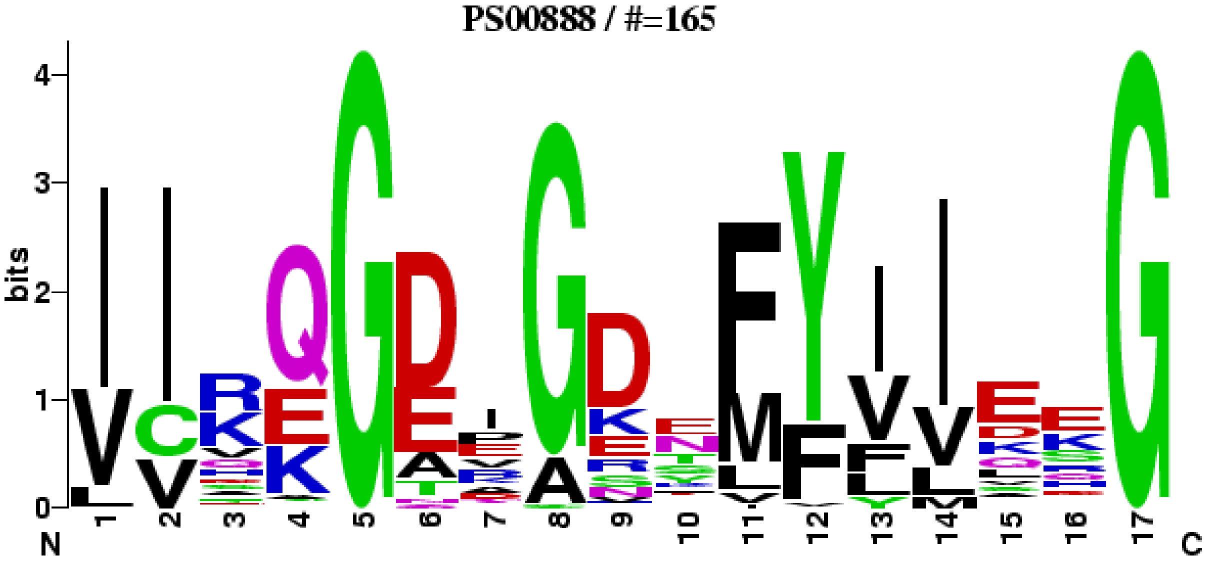

biological process
localization
single-organism process
cellular process
establishment of localization
transport
single-organism localization
single-organism cellular process
single-organism transport
ion transport
transmembrane transport
ion transmembrane transport

Other examples of HMM and profile use in protein analysis

- Predicting secondary structure
- Predicting transmembrane proteins and signal peptides
- Predicting functional motifs and posttranslational modifications (PROSITE database)

Cyclic nucleotide-binding domain signature 1:

`[LIVM]-[VIC]-x-{H}-G-[DENQTA]-x-[GAC]-{L}-x-[LIVMFY](4)-x(2)-G`



Oznamy

- Deadline of HW2 extended until Dec. 7
- HW3 will be published next week
- Next Thursday Dec.2: lecture and tutorials cancelled
- Thursday Dec.9: lecture and tutorials online
- Thursday Dec.16:
  - optional presentations of journal club during lecture time
  - tutorial for comp.sci. will take place
  - tutorial for biologists possibly cancelled
- End of semester deadlines
  - HW3 Tuesday Dec. 14, journal club reports Friday Dec. 17
- On Thursday Dec. 9, we will discuss:
  - if you want to present journal club (discuss in the group)
  - date of the exam (bring dates of other exams)

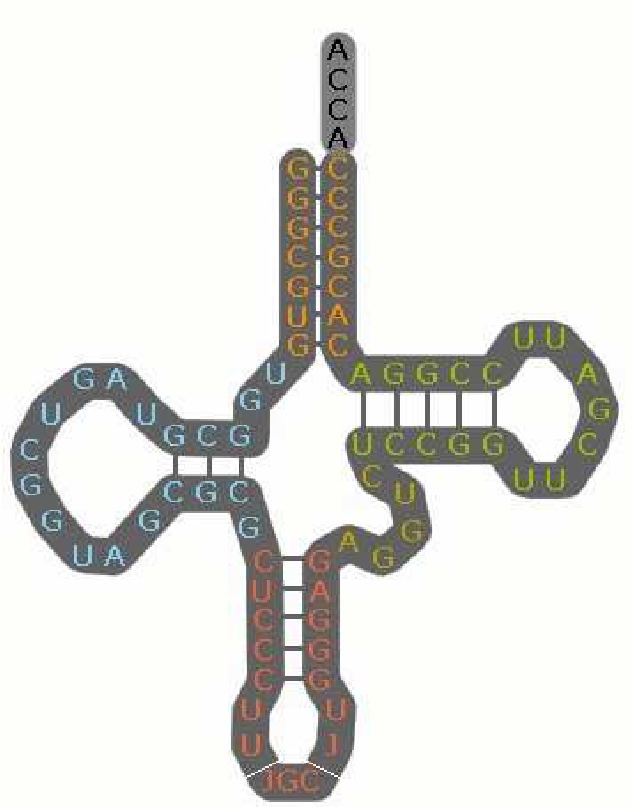
Recall: journal club report

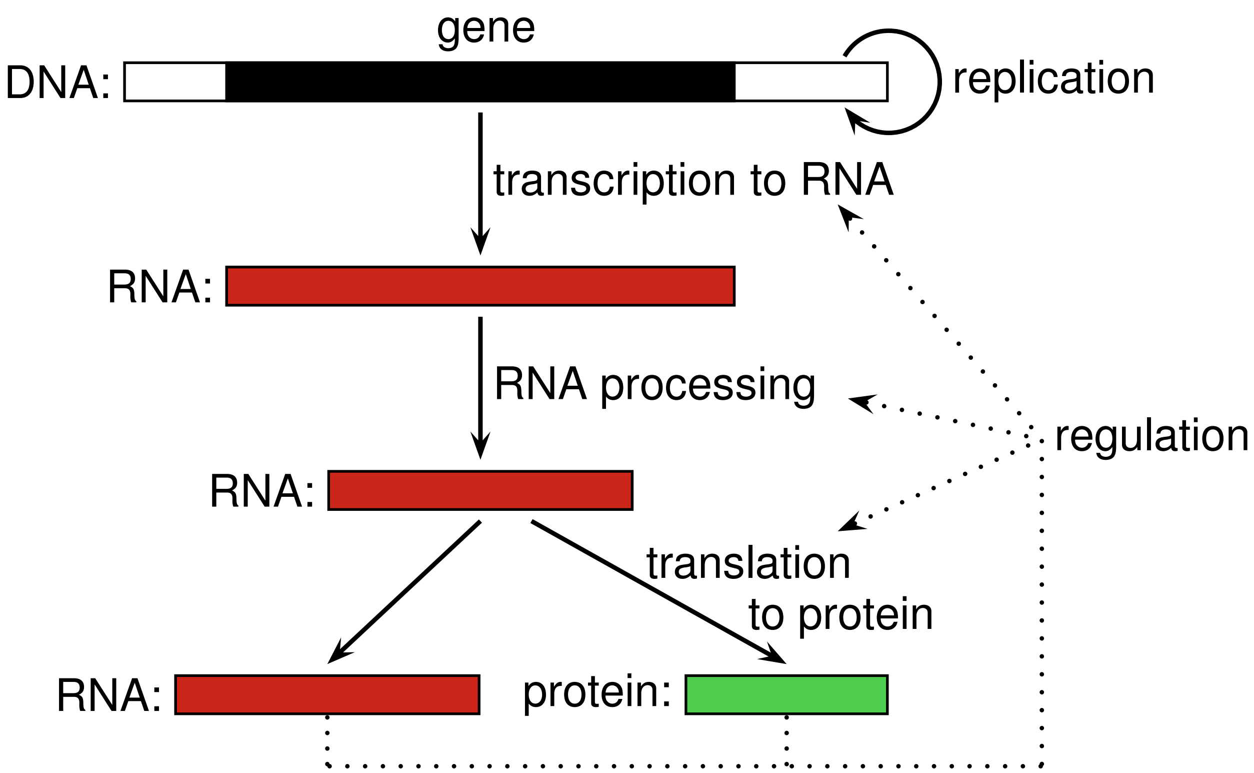
- The main methods and results of the article in your own words
- Understandable for students of this course (both comp.sci. and bio)
- You do not have to cover the entire content of the article in the report and, conversely, you can use other resources
- Try to express your own view of the topic, do not strictly follow the text of the article
- The recommended length is about 1-2 pages per person, one coherent text
- The report should list the members of the group who have actively participated. They will get the same points (the rest zero)
- Submit via Moodle, 1 pdf per group

RNA

Tomáš Vinař

Nov. 25, 2021




gene
DNA:
replication
transcription to RNA
RNA:
RNA processing
regulation
RNA:
translation
to protein
RNA:
protein:

Properties of RNA

Differences from DNA

- contains ribose instead of deoxyribose
- contains uracyl instead of thymine (bases A,C,G,U)
- single-stranded molecules, usually shorter
- complex secondary structure with paired complementary regions
- pairs A-U, C-G as well non-canonocal pairs e.g. G-U
- various functions in the cell:
  central role in gene expression (messanger RNA, transfer RNA, ribosomal RNA),
  regulation of expression,
  catalythic functions,
  transfer of genetic information for RNA viruses

RNA structure

Example: transfer RNA

Secondary structure:

pairing of nucleotides

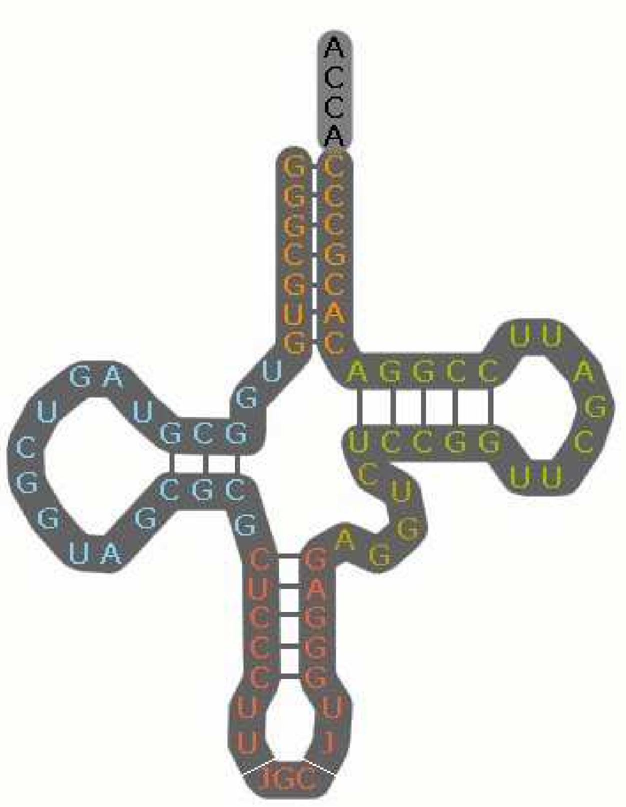
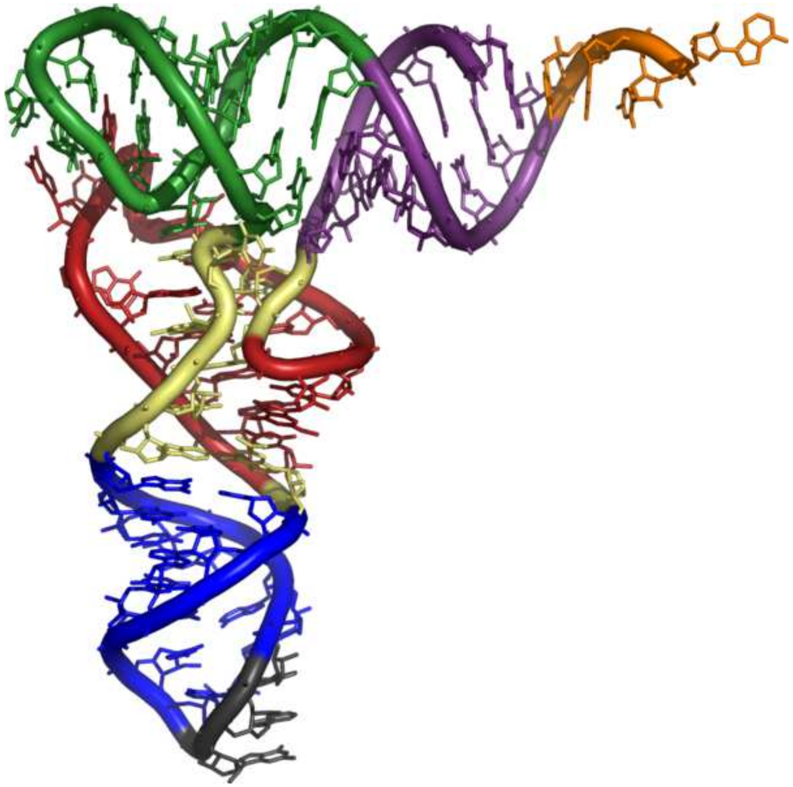


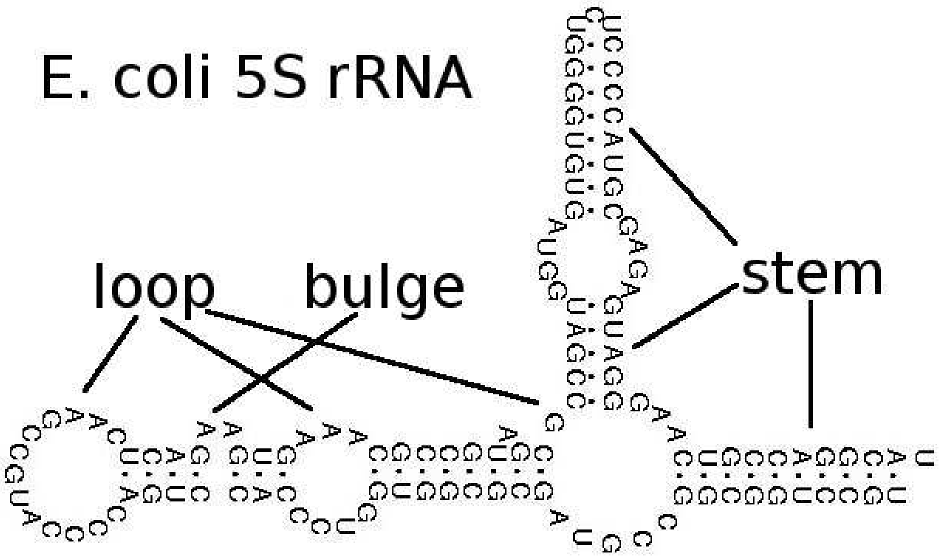
Figure source: Wikipedia

Tertiary structure:

3D coordinates



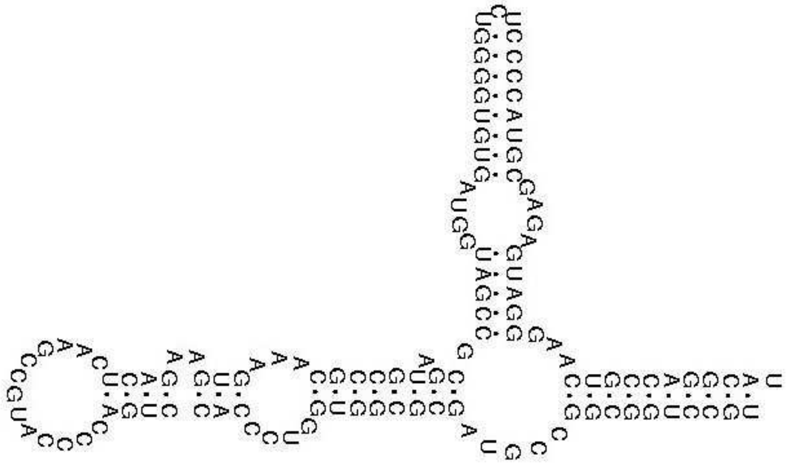
Sekundárna štruktúra RNA

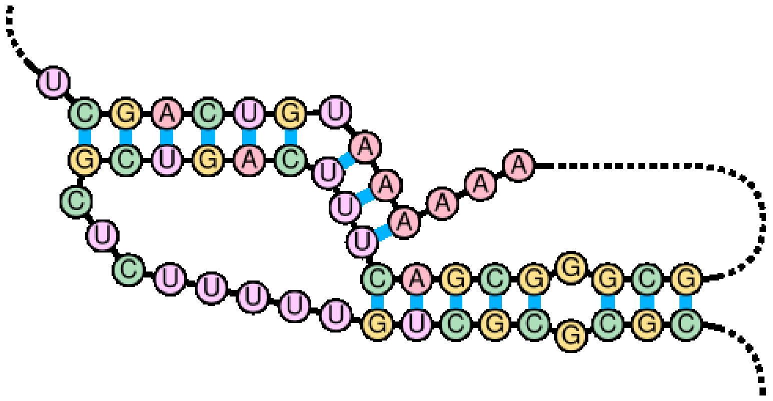


Representation using well-parenthesized expression:

```
((((((((((.....(((   ).)).((   ))...)))))))))).
UGCCUGGCGGCCGUAGCG...UAGCGCC...GGGAACUGCCAGGCAU
```

Well-parenthesized expression vs. pseudoknots





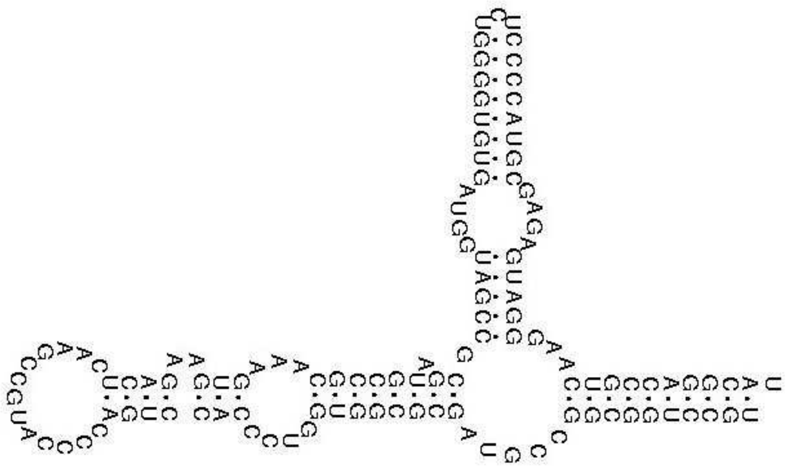
Left: can do well-parenthesized expression

```
((((((((((.....(((   ).)).((   ))...)))))))))).
UGCCUGGCGGCCGUAGCG...UAGCGCC...GGGAACUGCCAGGCAU
```

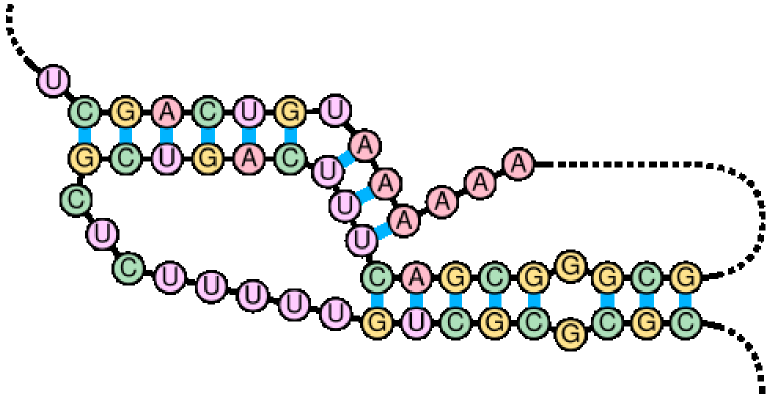
Right: pseudoknot (cannot do well-parenthesized expression)

```
.((((((.(((...[[[.[[[[[))))))))).   .]]]]].]]]
UCGACUGUAAAAAAGCGGGCGACUUUCAGUCGC...UGUCGCGCGC
```

Well-parenthesized expression vs. pseudoknots



without pseudoknots

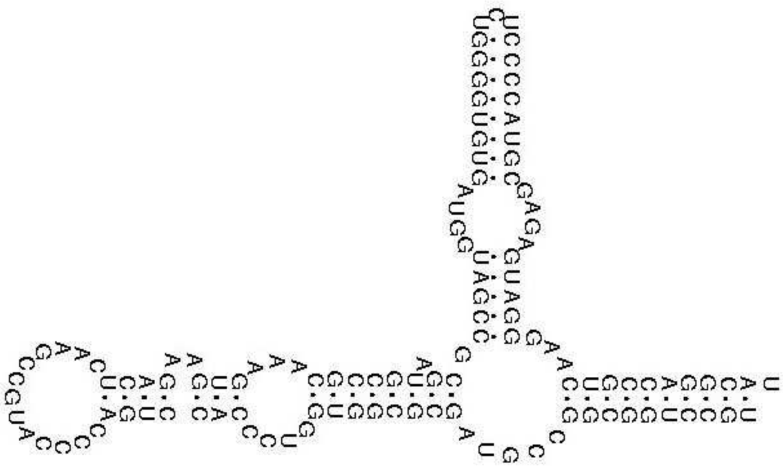


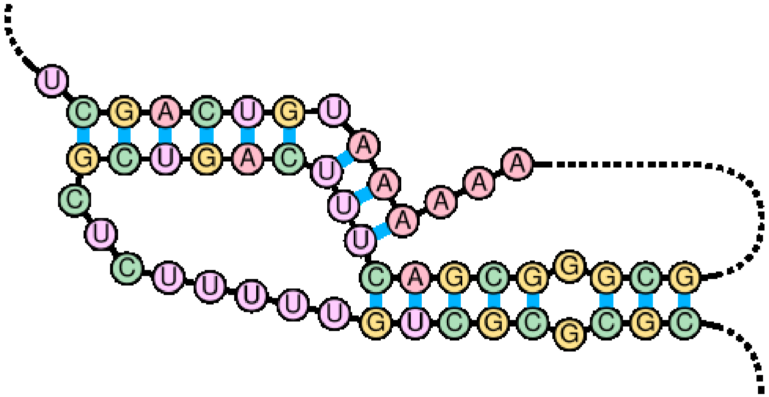
pseudoknot

Approx. 1.4% of paired RNA bases involved in pseudoknots

Yet many algorithms **ignore pseudoknots**

Well-parenthesized expression vs. pseudoknots

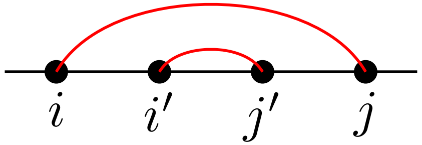




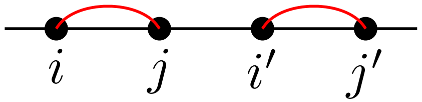
Mathematical structure of secondary structure w/o pseudoknots:

If position i is paired with j and position i' with j' where $i < i'$ then either $i < i' < j' < j$ or $i < j < i' < j'$.

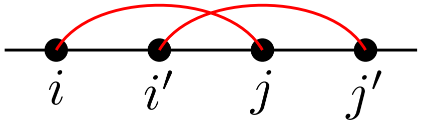
good:



good:



bad:



Problem: determining secondary RNA structure

Input: RNA sequence
Goal: find which bases are paired

Simplified formulation: find well-parenthesized expression corresponding to the structure with the highest number of complementary pairs A-U, C-G.

Example:

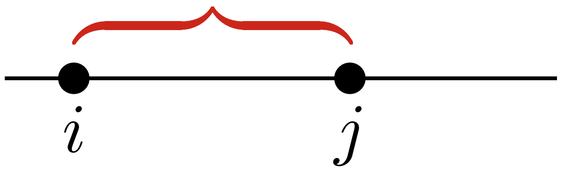
```
Input:   GAACACAUGUAAAAUUUGUC
Output:  ((.(((()))(((.))))))
```

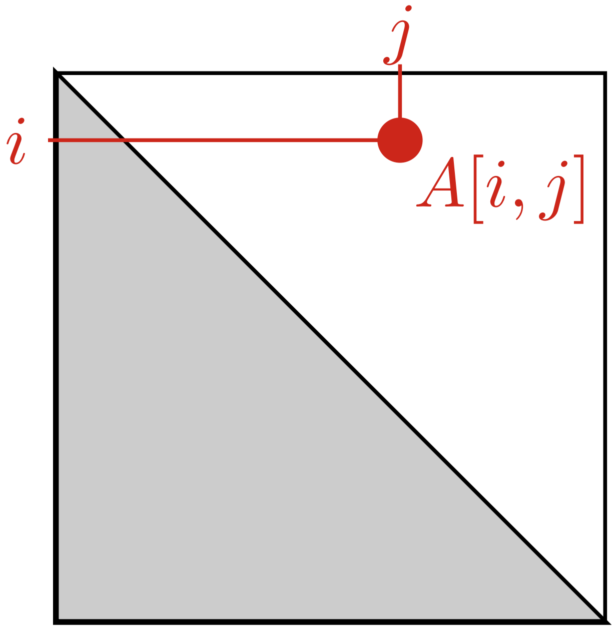
Nussinov algorithm

Dynamic programming:

Given RNA $x_1, \ldots, x_n$.

$A[i,j] =$ the maximum number of matched pairs in $x_i, x_{i+1}, \ldots, x_j$





Nussinov algorithm

Dynamic programming:

Given RNA $x_1, \ldots, x_n$.

$A[i, j]$ = the maximum number of matched pairs in $x_i, x_{i+1}, \ldots, x_j$

Recurrence:

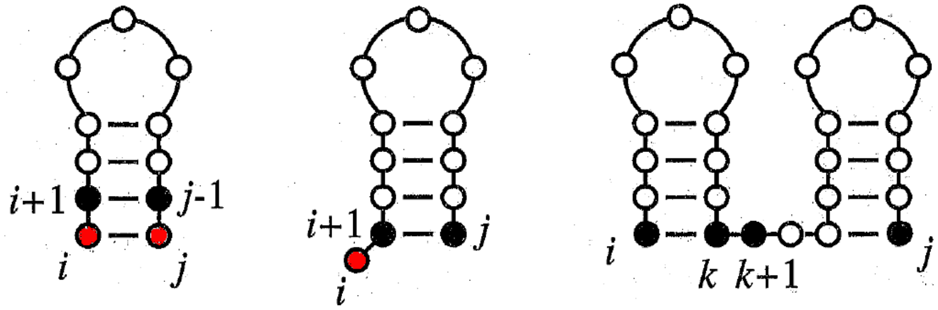
Substrings of length 1: no pairs possible $\Rightarrow A[i, i] = 0$

Longer substrings:

– x_i not involved in a pair: $A[i, j] = A[i+1, j]$

– x_i paired with x_j: $A[i, j] = A[i+1, j-1] + c(x_i, x_j)$

– x_i paired with x_k $(k < j)$: $A[i, j] = A[i, k] + A[k+1, j]$

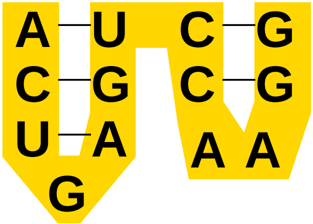


Rekurencia: $A[i,j] = \max \begin{cases} A[i+1,j], \\ A[i+1,j-1] + c(x_i,x_j), \\ \max_{k=i+1\dots j-1}\{A[i,k] + A[k+1,j]\} \end{cases}$

	A	C	U	G	A	G	U	C	C	A	A	G	G
A	0	0	1	1	1	2	3	3	3	3	3	4	5
C		0	0	1	1	2	2	2	2	3	3	4	4
U			0	0	1	1	1	2	2	3	3	3	3
G				0	0	0	1	2	2	2	2	3	3
A					0	0	1	1	1	1	1	2	3
G						0	0	1	1	1	1	2	2
U							0	0	0	1	1	1	2
C								0	0	0	0	1	2
C									0	0	0	1	1
A										0	0	0	0
A											0	0	0
G												0	0
G													0

$c(x_i,x_j) = \begin{cases} 1 & \text{if } x_i\text{-}x_j \text{ is A-U or C-G pair} \\ 0 & \text{otherwise} \end{cases}$

$A[i,j] = 0$ for $i \geq j$



Complexity:

$O(n^3)$ time

$O(n^2)$ memory

Minimum free energy (MFE) folding

More realistic formulation

Assumption: the molecule in the state of equilibrium with minimum Gibbs free energy.

Energies for modules measured experimentally.

Nearest neighbor model: parameters = energies for neighbouring pairs in helixes, lengths of loops, etc.

Derived from experimental measurements.

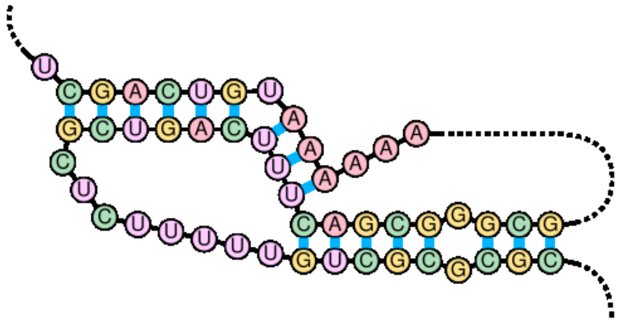
Example:

```
5’  Cx  3’
3’  Gy  5’
```

y:	A	C	G	U
x:A	.	.	.	-2.1
C	.	.	-3.3	.
G	.	-2.4	.	-1.4
U	-2.1	.	-2.1	.

Algorithms similar to the Nussinov algorithm [Zuker and Stiegler, 1981].

Algorithms allowing pseudoknots



NP-hard in general [Lyngso and Pedersen, 2000].

Slow dynamic programming $O(n^4)$ – $O(n^6)$ for certain pseudoknot types [Rivas and Eddy, 1999].

Or use heuristics [Ren et al., 2005] (repeated greedy formation of strong helixes).

Probabilistic models for RNA secondary structure prediction

Want: Generative model for pairs sequence, secondary structure

Use: For a given sequence, find most probable structure

HMMs are **not** suitable: cannot capture dependencies between distant pairs

Solution: **Stochastic context-free grammars (SCFGs)**

- extension of context-free grammars
- individual rules will get probabilities

Stochastic context-free grammars (SCFGs)

non-terminals (upper-case) similar to states in HMMs

terminals (lower-case) represent nucleotides

rules rewrite non-terminals to strings of terminal and non-terminals

each rule has assigned probability

Example: single non-terminal, 14 rules (ϵ =empty string)

$$S \to \overbrace{aSu}^{0.1} \mid \overbrace{uSa}^{0.1} \mid \overbrace{cSg}^{0.1} \mid \overbrace{gSc}^{0.1} \mid$$

$$\overbrace{aS}^{0.05} \mid \overbrace{cS}^{0.05} \mid \overbrace{gS}^{0.05} \mid \overbrace{uS}^{0.05} \mid \overbrace{Sa}^{0.05} \mid \overbrace{Sc}^{0.05} \mid \overbrace{Sg}^{0.05} \mid \overbrace{Su}^{0.05} \mid \overbrace{SS}^{0.1} \mid \overbrace{\epsilon}^{0.1}$$

In each step choose the left-most non-terminal

rewrite with a randomly chosen rule:

$$S \to SS \to aSuS \to acSguS \to acuSaguS \to acugSaguS \to$$

$$acugaguS \to acugagucSg \to acugaguccSgg \to acugaguccSagg \to$$

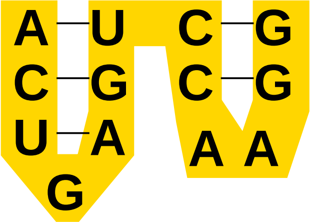
$$acugaguccaSagg \to acuguguccaagg$$

Stochastic context-free grammars

Example:

$S \to aSu|uSa|cSg|gSc|aS|cS|gS|uS|Sa|Sc|Sg|Su|SS|\epsilon$

$S \to SS \to aSuS \to acSguS \to acuSaguS \to acugSaguS \to$
$acugaguS \to acugagucSg \to acugagucgScg \to acugagucgSacg \to$
$acugagucgaSacg \to acugugucgaacg$



Problem: Find most probable derivation of given RNA

Bases generated in a single rule represent **paired bases**

Solution: Dynamic programming, algorithm CYK, $O(n^3)$ time

Training: Probabilities trained from known RNA structures

Grammars vs. energy minization

Grammar advantages:

- parameters can be trained automatically, no expensive experiments
- can be extended to multiple sequences

Grammar disadvantages:

- simple grammars do not capture full complexity of the problem
- lower accuracy

RNA sequence evolution

Often correlation between mutations in paired bases
e.g. C changes to A, paired G changes to U simultaneously

Example: several sequences from t-RNA D-arm

```
((((.............))))
GCUCAGCC.CGGG...AGAGC
GCCUAGCC.UGGUCA.AGGGC
GUCUAGC...GGA...AGGAU
GAGCAGUU.CGGU...AGCUC
GUUCAAUC..GGU...AGAAC
```

Problem: given a multiple alignment of RNA sequences
find a common RNA structure

(common structure will exhibit correlations between paired bases)

Common RNA structure for several RNA sequences

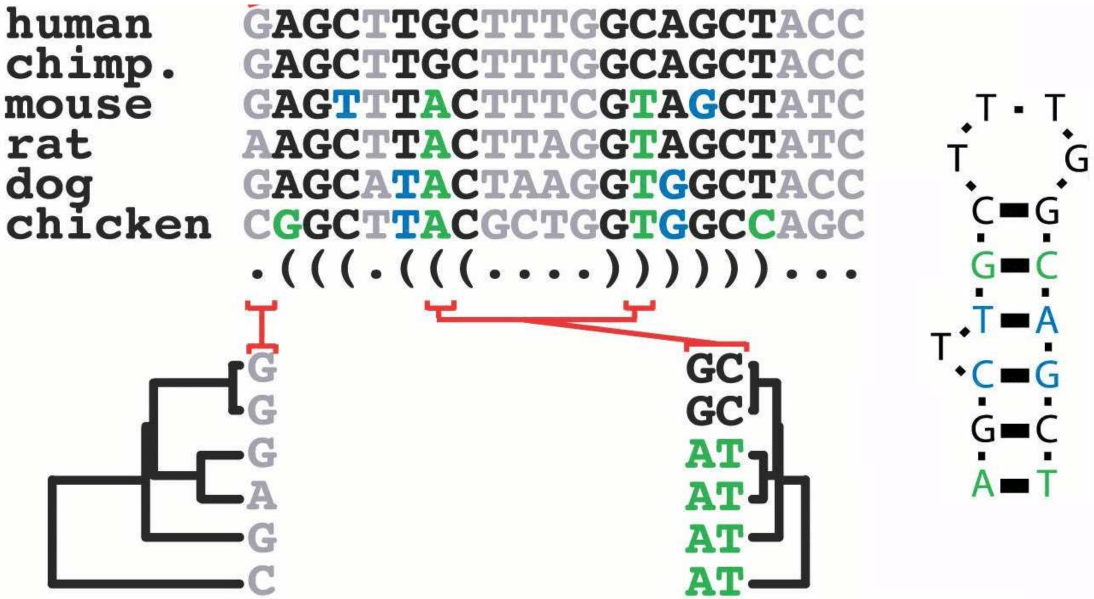
Phylo-SCFG:

- terminals will be **whole alignment columns**

  use phylogenetic tree structure

- unpaired bases emitted using a regular substition matrix

- paired bases emitted using a 16×16 substition matrix (all pairs)

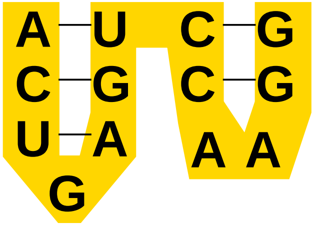


```
human    GAGCTTGCTTTGGCAGCTACC
chimp.   GAGCTTGCTTTGGCAGCTACC
mouse    GAGTTTACTTTCGTAGCTATC
rat      AAGCTTACTTAGGTAGCTATC
dog      GAGCATACTAAGGTGGCTACC
chicken  CGGCTTACGCTGGTGGCCAGC
         .(((.(((....))))))...
```

Problem: Finding known types of RNA genes in genomes

- Rfam database contains structures for > 4000 RNA families represented using probabilistic models
- Similar idea to profile HMMs used for representation of protein families (Pfam database)
- Special type of SCFGs called **covariance models**

Covariance models (CMs)



$$\begin{array}{lll}
S \to B_1 & P_1 \to aP_2u & P_4 \to cP_5g \\
B_1 \to P_1P_4 & P_2 \to cP_3g & P_5 \to gL_2c \\
 & P_3 \to uL_1a & L_2 \to aL_3 \\
 & L_1 \to gE_1 & L_3 \to aE_2 \\
 & E_1 \to \epsilon & E_2 \to \epsilon
\end{array}$$

- S =start, E_i =end, P_i =pair,

  L_i =unpaired base on the left, R_i =unpaired base on the right

  other non-terminals to represent indels

- terminals (bases) emitted with probabilities **specific to each alignment column**

  e.g. $P_1 \to \overbrace{aP_2u}^{0.2} \,|\, \overbrace{uP_2a}^{0.2} \,|\, \overbrace{cP_2g}^{0.4} \,|\, \overbrace{cP_2u}^{0.1}$

Covariance models (CMs)

Uses:
finding occurrences of a gene in DNA (local alignment),
finding structure of a new gene from a known family (global alignment).

Dynamic programming: time $O(MND^2)$
$M =$ the number of non-terminals, proportional to the alignment length
$N =$ the length of DNA ,
$D =$ max. length of an RNA gene (related to M).

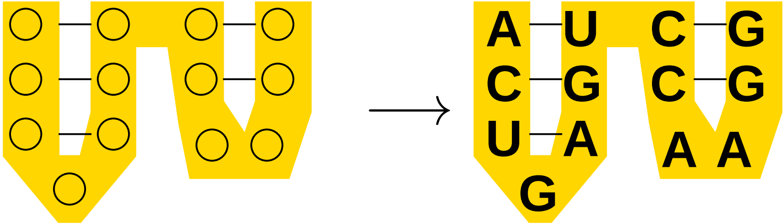
Heuristic speedup: find potential sites with sequences similar to known family representatives, apply CM only there

Problem: RNA secondary structure design

Given RNA secondary structure (pairing)

Find a sequence for which this is the optimal structure.

No known efficient algorithm, but fast heuristics work well



Use: research on possible RNA structures, drug design (ribozymes, riboswitches), RNA for laboratory techniques, RNA nanostructures

Summary

- RNA secondary structure prediction:
  energy minimization, probabilistic SCFGs
- Can achieve better results if we use a multiple alignment of several RNA sequences with a common structure (PhyloSCFG)
- Known RNA families can be represented by covariance models, these can be used to locate occurrences in novel sequences
- Rfam database
- Most problems can be solved by dynamic programming
  - somewhat slow
  - ignores pseudoknots
- Other interesting problems: RNA design

Announcements

- Today last lecture, afterwards tutorial for biologists
- Next Thursday Dec.16.:
  – last tutorial for comp.sci.
  – optional presentations of journal club during lecture time (interest?)
  – tutorial for biologists possibly cancelled
- End of semester deadlines
  – journal club reports Friday Dec. 17
  – HW3 Tuesday Dec. 21

Exam (comp.sci. only)

The main part is **written:**

- You need at least 50% of points
- Time 3 hours
- About 50% of points for simple questions,
  – examples will be on the course website
  – in case of interest tutorial session before exam
- The rest of the questions mostly designing/modifying an algorithm or model
- **Date?**
- Online or in person, depending on circumstances
- You can use pen, simple calculator
  and a cheat sheet up to 2 A4 two-sided sheets

Written exam, online version (comp.sci. only)

- Exam questions and submission in Moodle
- MS teams: annoucements, questions
- Write in an editor, create pdf
  or write on paper, scan/photo, convert to pdf
- Allowed aids:
  Same as in person (incl. cheat sheet)
  Text and image editors, software for digitization of handwritten pages
  MS Teams to communicate with instructors
  Moodle for getting and submitting exam
- Not allowed:
  Communication with other persons except instructors
  Other webpages
  Other software (e.g. specialized bioinformatics programs, compilers)

Oral exam

- Only for online exam
- Videocall in MS Teams
- After written exam, time slots over several days
- We will discuss your exam
- You should be able to explain your answers in detail
- Oral exam influences exam grade
- If you are unable to explain your answers, you will get Fx

“Second chance” exam: the same for as the first or oral-only the dates arranged with those who need them

Population Genetics

Broňa Brejová

December 9, 2021

Population genetics

- Genomes of different individuals of the same species differ
- These differences cause differences in phenotype (appearance, behaviour, diseases,. . .)
- We can sequence multiple individuals and compare with reference sequence

Possible applications:

- Impact of individual genetic differences
- History and structure of populations (subpopulations, migration, historical changes in size)

SNPs (Single Nucleotide Polymorphisms)

- SNP: a single base mutation (present in $> 1\%$ individuals)
- Usually only two forms : **major** and **minor** allele
- Small change at some places in the genome can cause large phenotypic changes

Systematic mapping of SNPs:

1000 Genomes Project 2008-2015
identify 95% of SNPs with 1% minor allele frequency
using next generation genome sequencing

Trait/Disease Association Mapping

- Traits and diseases emerge by the combination of genetic and environmental influences
- Goal: Identify genetic influences.
  - Disease mechanisms?
  - What is the risk of inheritance?
  - How can we design and target new drugs (pharmacogenomics)?
    E.g. mutations of cytochrome family P450 genes
    influence metabolism of drugs in the liver,
    thus influence necessary dose

Diploid genomes

- Human has a **diploid genome**:
  each human cell contains two copies of chromosomes 1. . . 22
  plus sex chromosomes X,X or X,Y

- From each pair, one chromosome comes from mother and one from father

- For a SNP with alleles (forms) a and A,
  an individual is **homozygote** (aa or AA),
  or **heterozygote** (aA)

- A disease caused by allele a can appear only in homozygotes aa,
  or also in heterozygotes aA, or more severe for aa than aA

Diploid genomes

- Human has a **diploid genome**:
  each human cell contains two copies of chromosomes 1. . . 22
  plus sex chromosomes X,X or X,Y

- From each pair, one chromosome comes from mother and one from father

- For a SNP with alleles (forms) a and A,
  an individual is **homozygote** (aa or AA),
  or **heterozygote** (aA)

- **Haplotype:** combination of alleles of different SNPs on the same chromosome (inherited from one parent)
  Diploid individual has two haplotypes

chr1 from mother:	. . . A. . . T. . . G. . . . . .
chr1 from father:	. . . T. . . C. . . A. . . . . .

Testing a single SNP

Contingency table - the number of haplotypes

Dog size vs allele at chr15:44,228,468 [Sutter et al., 2007]

	allele A	allele a	total
small dog (< 9 kg)	14	535	549
large dog (> 31 kg)	339	38	377
total	353	573	

Test if columns and rows are **independent (null hypothesis)**

If null hypothesis **rejected**, there is association between SNP and size (not necessarily causal)

If null hypothesis **not rejected**, association not found (perhaps will be found with more data)

Testing independence in a contigency table

	allele A	allele a	total
small dog	14	535	549
large dog	339	38	377
total	353	573	926

Fisher's exact test: (Fisher's exact test) exact probability from hypergeometric distribution

χ^2 test (chí-kvadrát): popular approximate test, appropriate for large counts

In practice also more complex statistical methods / models
(diploid genome, family relationships, . . .)

Testing independence in a contingency table by χ^2 test

	allele A	allele a	total
small dog	14	535	549
large dog	339	38	377
total	353	573	926

Under null hypothesis (independence of rows ans columns):

$\Pr(A) = 353/926 = 0.381,\ \Pr(a) = 0.619$

$\Pr(s) = 549/926 = 0.593,\ \Pr(l) = 0.407$

$\Pr(A, s) = \Pr(A)\Pr(s) = 0.226$

$\Pr(a, s) = \Pr(a)\Pr(s) = 0.367$

$\Pr(A, l) = \Pr(A)\Pr(l) = 0.155$

$\Pr(a, l) = \Pr(a)\Pr(l) = 0.252$

Under the null hypothesis we expect 926 haplotypes in the table divided in ratios 0.226:0.367:0.155:0.252

Testing independence in a contingency table by χ^2 test

Real table

$O_{i,j}$ (observed):

	A	a	total
small	14	535	549
large	339	38	377
total	353	573	926

Expected under null

$E_{i,j}$ (expected):

	A	a	total
small	209.3	339.8	549
large	143.5	233.4	377
total	353	573	926

Compute $\chi^2 = \sum_{i \in \{s,l\}} \sum_{j \in \{A,a\}} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$

$\chi^2 = (14 - 209.3)^2/209.3 + (535 - 339.8)^2/339.8 + (339 - 143.5)^2/143.5 + (38 - 233.4)^2/233.4 = 724.3$

χ^2 is a measure of difference between tables O and E.

Always $\chi^2 \geq 0$, and $\chi^2 = 0$ only if tables equal.

Testing independence in a contingency table by χ^2 test

$O_{i,j}$ (observed):

	A	a	total
small	14	535	549
large	339	38	377
total	353	573	926

$E_{i,j}$ (expected):

	A	a	total
small	209.3	339.8	549
large	143.5	233.4	377
total	353	573	926

Compute $\chi^2 = \sum_{i\in\{s,l\}} \sum_{j\in\{A,a\}} \frac{(O_{i,j}-E_{i,j})^2}{E_{i,j}} = 724.3$

Under null hypothesis, χ^2 is approximately from $\chi^2(1)$ distribution, i.e. **chi squared with one degree of freedom**.

1 degree: if we know E and 1 number from O, the rest of O can be computed

The probability that under null we get by chance $\chi^2 \geq 724.3$ is $1.6 \cdot 10^{-159}$ (P-value)

To **reject null hypothesis** use threshold e.g. $P < 0.05$, $\chi^2 > 3.841$

Dependencies between two different SNPs

Consider SNP with alleles p/P and another with alleles q/Q.
Count haplotypes pq, PQ, pQ, Pq

Example: 2000 haplotypes (1000 individuals)

	Q	q
P	474	611
p	142	773

$\chi^2 = 184.78$, P-value $4.4 \cdot 10^{-42}$

Columns and rows not independent, dependency between the SNPs

Example 2: Similar ratios of counts, but only 30 haplotypes:

	Q	q
P	7	9
p	2	12

$\chi^2 = 3.0867$, P-value 0.07893

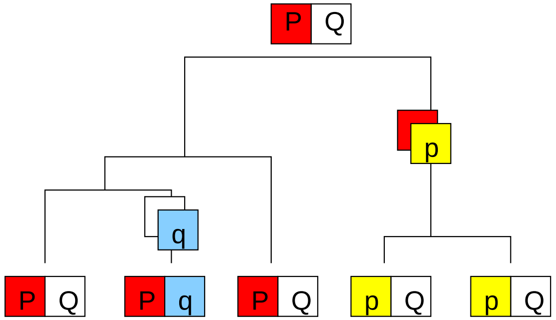
Null hypothesis not rejected for threshold P<0.05 ($\chi^2 > 3.841$)
Beware, χ^2 not appropriate for such low counts

Why are SNPs dependent?

SNPs on different chromosomes:

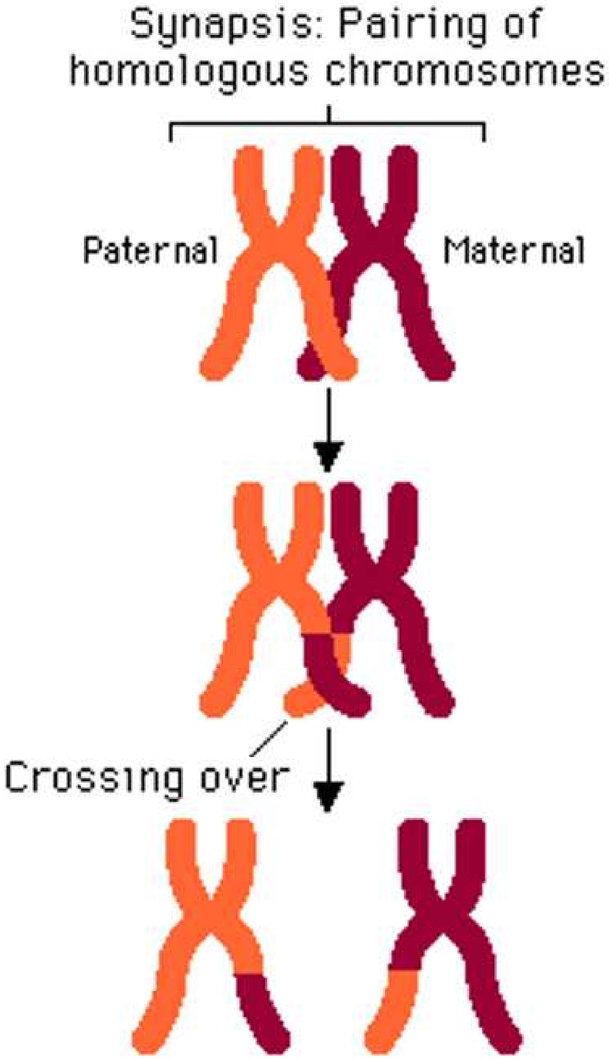
- Probabilities of individual alleles often independent
- $\Pr(pq) = \Pr(p)\Pr(q)$, $\Pr(PQ) = \Pr(P)\Pr(Q)$, etc.
- **linkage equilibrium (LE, väzbová rovnováha)**

SNPs nearby on the same chromosome:



- The same mutation happening twice is rare, recombination also relatively rare
- Allele combinations not completely random
- Correleation between SNPs
  $\Rightarrow$ **linkage disequilibrium (LD, väzbová nerovnováha)**

Recombination



Approx. 1-3 **recombinations** on 1 human chromosome during meiosis (production of sperm/eggs)

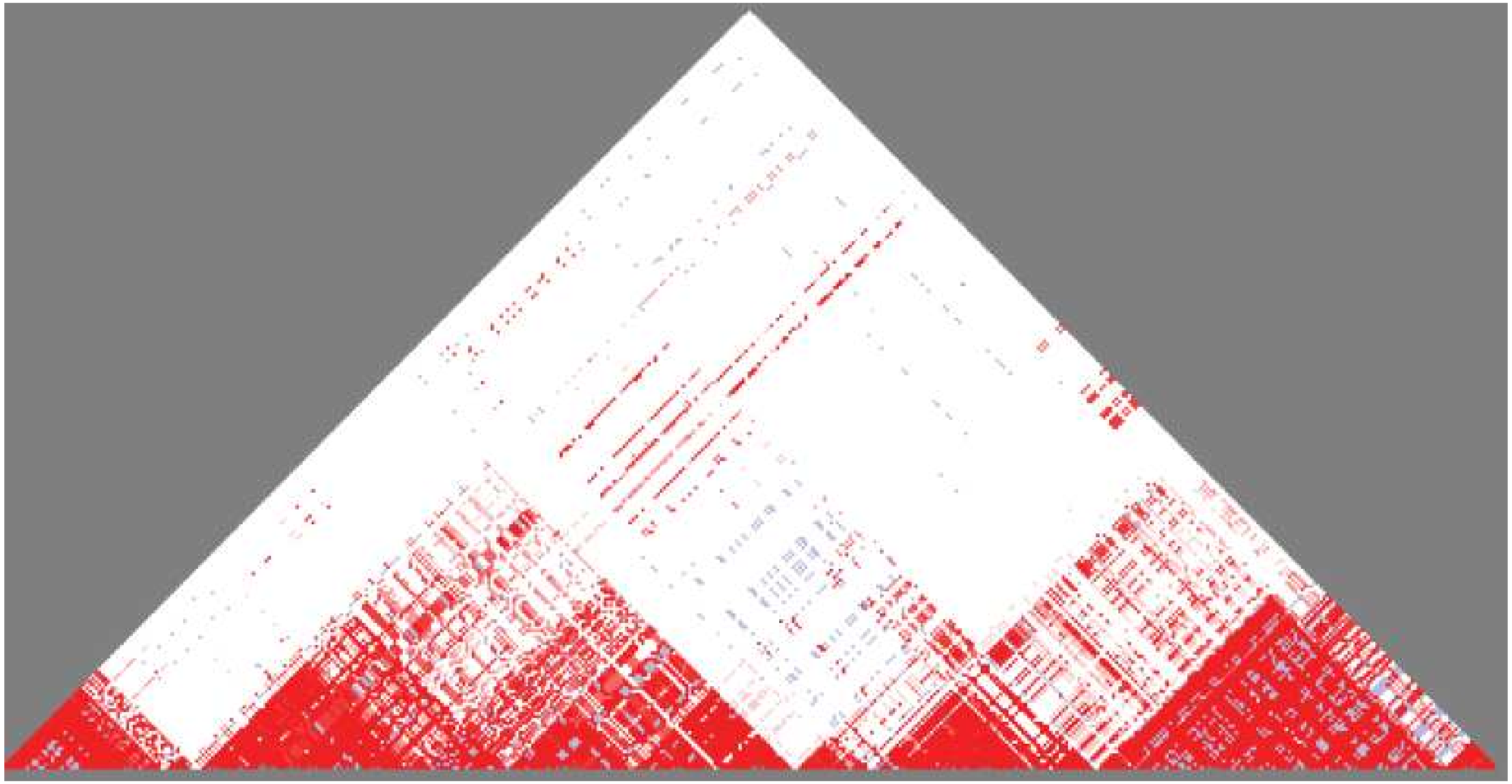
Recombination lowers LD

Assuming uniform recombination

- LD decreases with SNP distance on a chromosome
- LD decreases with SNP age
- Other factors: population structure, natural selection, recombination hotspots

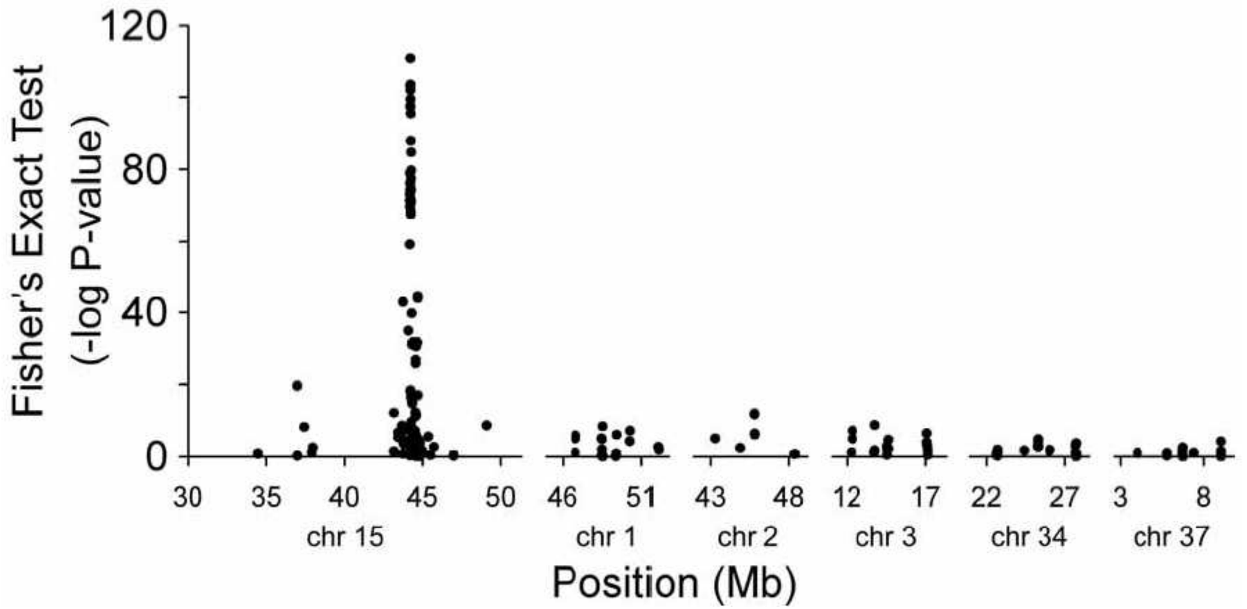
Linkage disequilibrium (LD) in the human genome

[The International HapMap Consortium, 2005]



Region ENm014 (500kB, chr 7), 90 people from Utah

Back to dogs: Whole-Genome Association Scan (WGAS)

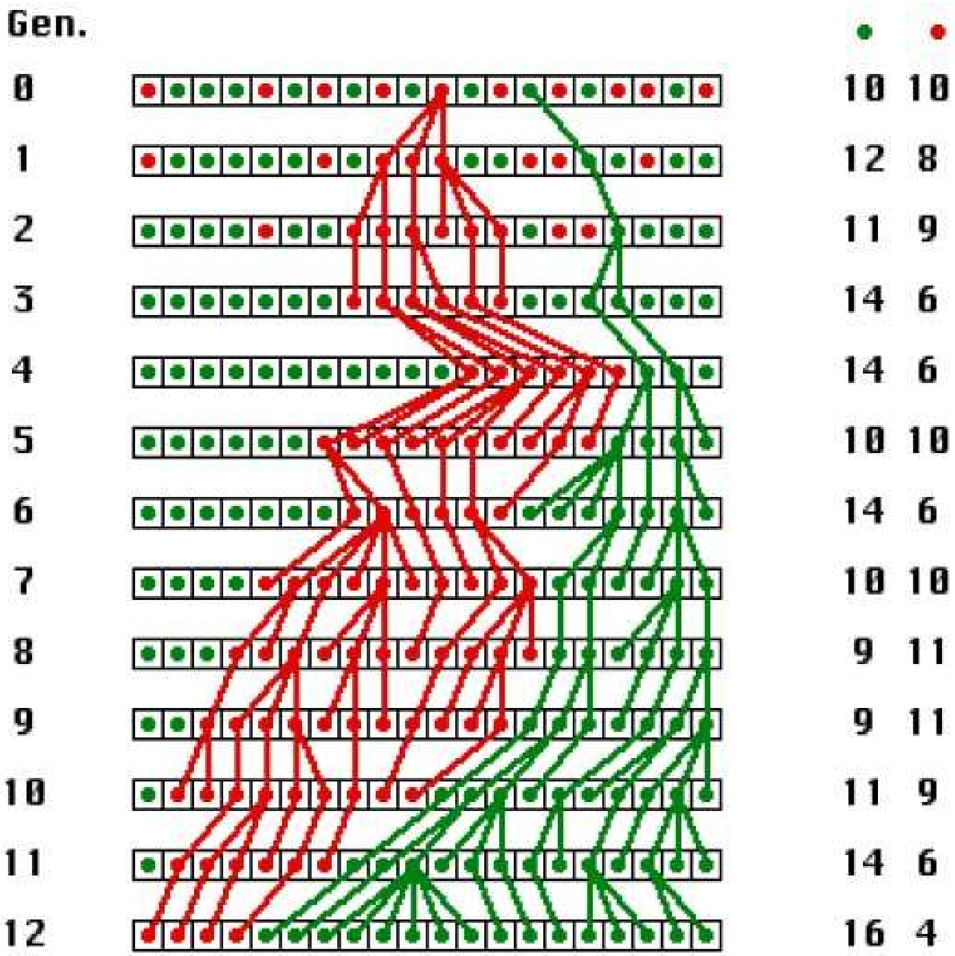


- For dog size, WGAS identified 84 kB region

- Causal SNP has to be more finely mapped by additional experiments

- **Large LD blocks** ⇒ only can identify large regions

Basic model of population genetics: Wright-Fischer model



Gen.	🟢	🔴
0	10	10
1	12	8
2	11	9
3	14	6
4	14	6
5	10	10
6	14	6
7	10	10
8	9	11
9	9	11
10	11	9
11	14	6
12	16	4

Lifecycle of SNPs in Wright-Fisher model

- Population of N haploid organisms
- One allele per organism (A or a)
- New generation created as a copy of a random parent (random mating), no influence of natural selection
- X_t: the count of allele a in generation t
- **Markov chain** with states $X_t \in \{0, 1, \ldots, N\}$

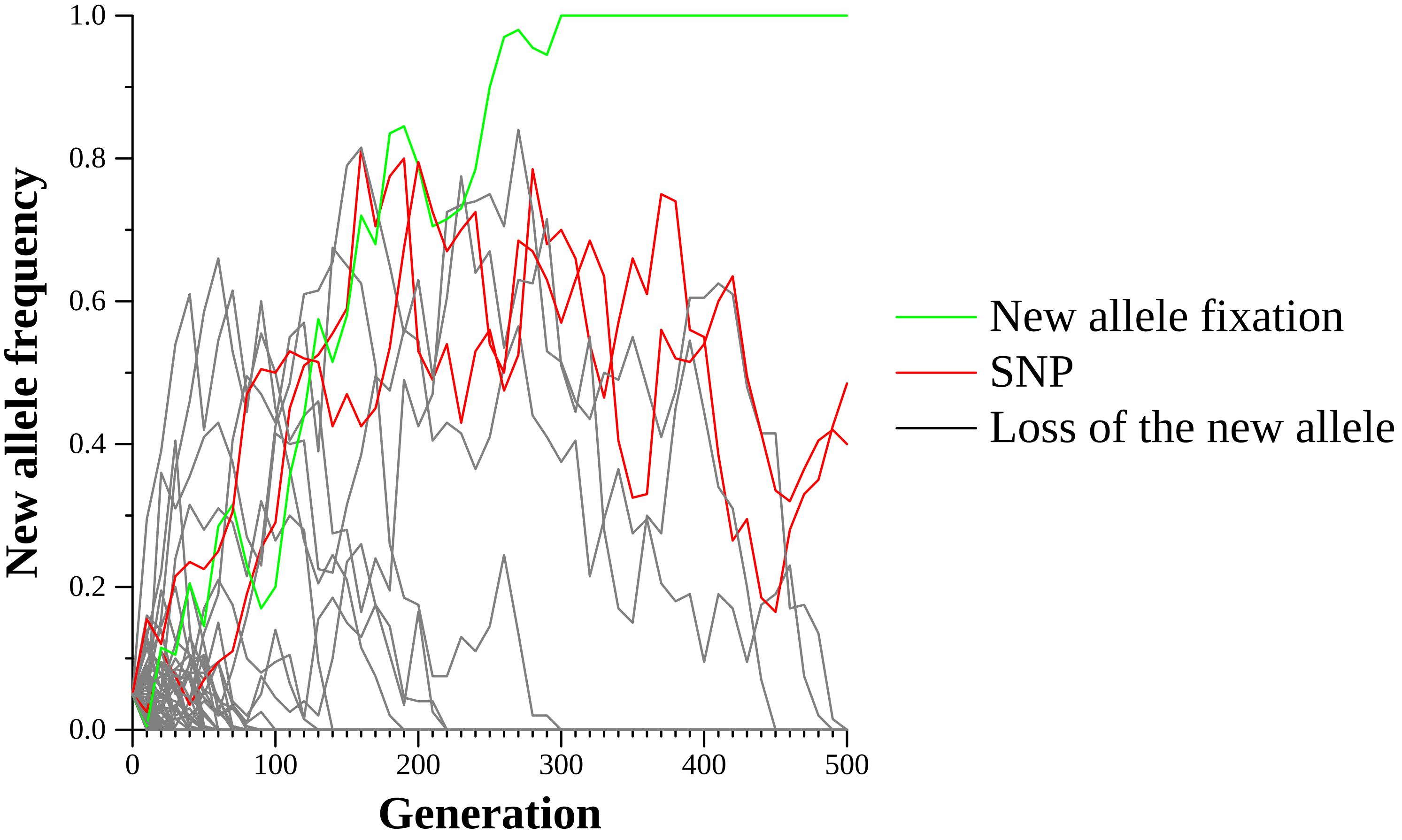
$$\Pr(X_t = j \,|\, X_{t-1} = i) = \left(\frac{i}{N}\right)^{j} \left(\frac{N-i}{N}\right)^{N-j} \binom{N}{j}$$

  (Probability that we have j copies of a in generation t, given i copies in generation $t-1$

- States 0 and N are **absorbing**

Random genetic drift

$N = 200$, $X_0 = 10$, 500 generations



More complex models of population

- **Mutations** introduce new alleles, these get eliminated or fixed by random genetic drift
- Speed of fixation influenced by **population structure** or **natural selection**.
- $\Rightarrow$ More complex probabistic models.

Analysis of population history using probabilistic models

Typical model parameters:

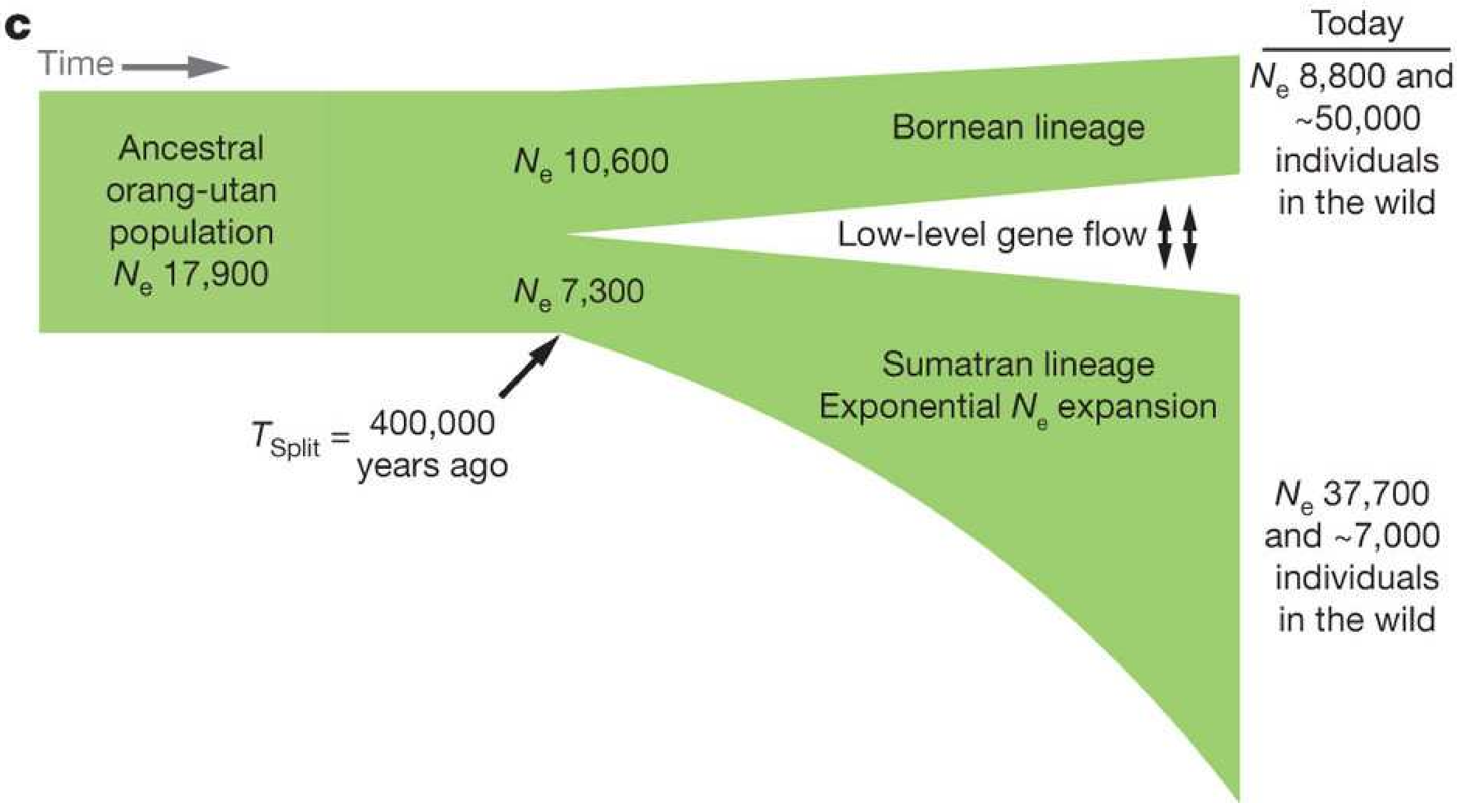
- efective population size
- frequencies of mutation and recombination

These parameters influence observed data:

- SNP frequencies (frequency of minor allele)
- Heterozygocity in diploid individuals
- The number and size of LD blocks

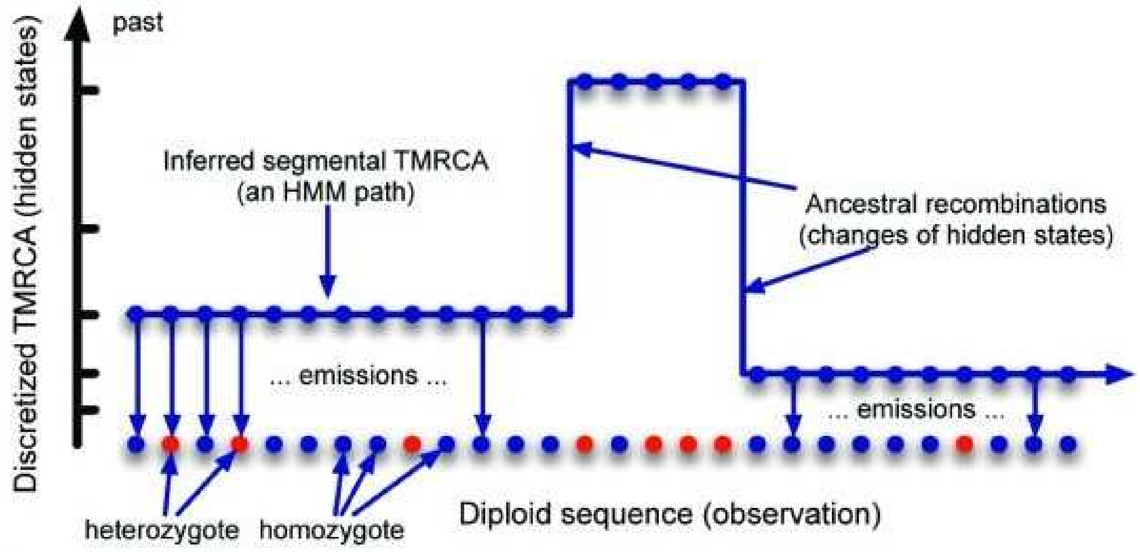
Standard approach: Find parameters of the model best explaining observed data in sequenced individuals

Example: Population history of orangutans


c
Time
Ancestral orang-utan population N_e 17,900
N_e 10,600
N_e 7,300
T_{Split} = 400,000 years ago
Bornean lineage
Low-level gene flow
Sumatran lineage Exponential N_e expansion
Today
N_e 8,800 and ~50,000 individuals in the wild
N_e 37,700 and ~7,000 individuals in the wild

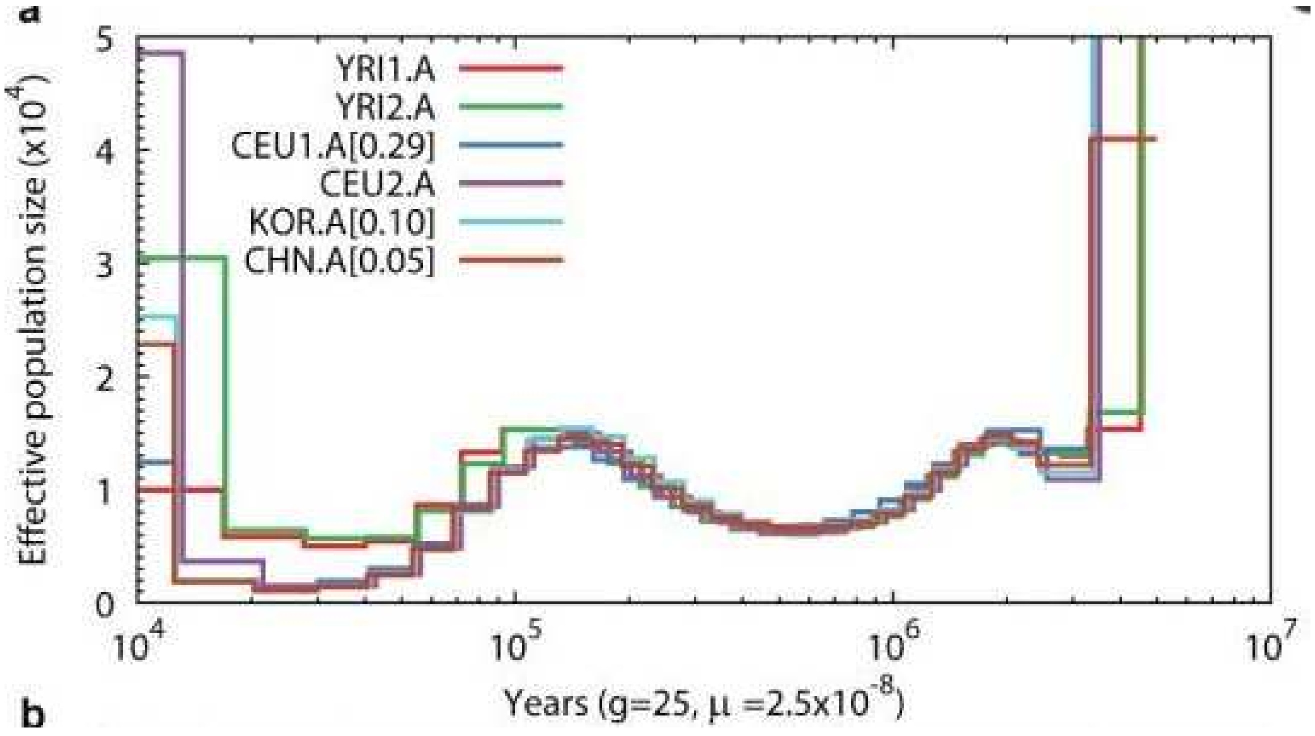
History of a human population from a single human genome (Li, Durbin 2011)

- **Model parameters:** effective human population time changing over time
- **Observed data:**
  - sizes of recombination blocks
  - distribution of time to the most recent common ancestor (TMRCA)



History of a human population from a single human genome

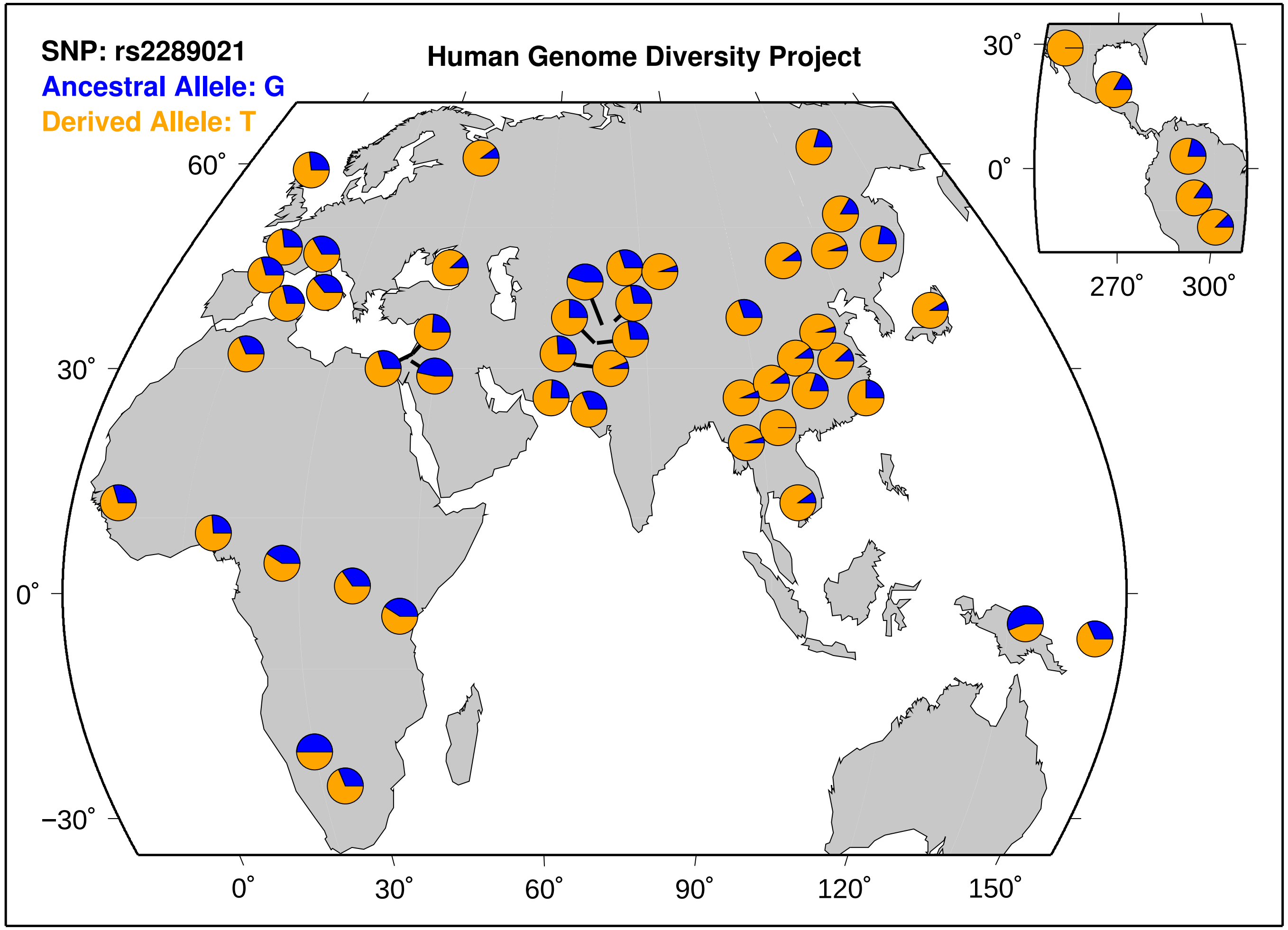
Task: Find historical population sizes best explaining observed statistics



Population structure

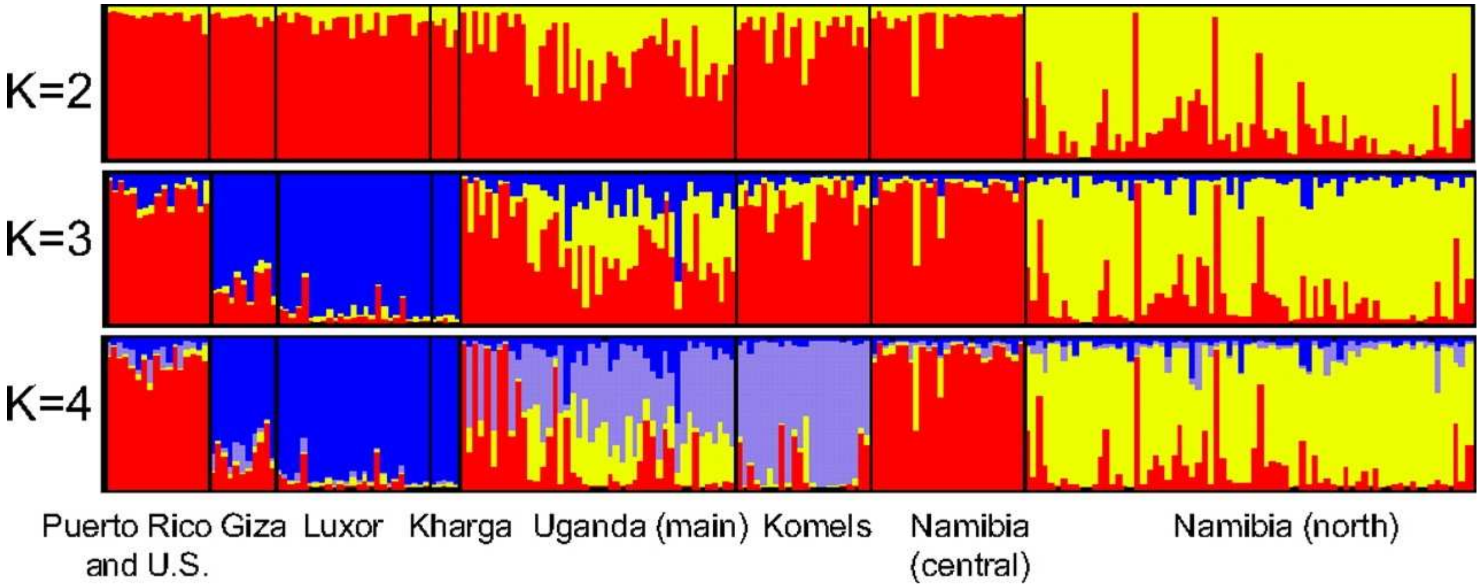
- Assumption so far: new generation produced by random mating
- Most organisms evolve in **subpopulations**, with limited migration between populations
- Frequencies of the same SNP in two different populations can be very different
- ⇒ “false” long-range correlations between SNPs (e.g., even between chromosomes) if we work with a mix of subpopulations
- ⇒ erroneous results in WGAS, LD studies, etc.

Example: allele frequencies of a particular SNP in different regions



from genome.ucsc.edu

Wild dog population structure



Boyko et al. PNAS 2009; software STRUCTURE Pritchard et al. Genetics 2000

- Program STRUCTURE splits population into K subpopulations (colors)
- Each column represents an individual from the population
- Ratio of colors represents ratio of SNPs in the mixture of the K subpopulations.

Algorithm used in STRUCTURE

- **Input:** Set of haplotypes X, which we want to separate into K subpopulations
- Define probabilistic model with the following variables:
  - $P_{i,j}$ - frequency of SNP j in subpopulation i
  - $Z_{i,j}$ - assignment of subpopulation to SNP j in haplotype i
  - Q_i - what portion of SNPs in haplotype i belong to which subpopulation
- Model defines $\Pr[X \mid P, Q, Z]$ and prior distribution for P, Q
- **Output:** $E[Q \mid X]$

Algorithm Markov Chain Monte Carlo (MCMC)

- Variables:
  - $P_{i,j}$ - frequency of SNP j in subpopulation i
  - $Z_{i,j}$ - assignment of subpopulation to SNP j in haplotype i
  - Q_i - what portion of SNPs in haplotype i belong to which subpopulation
- Start with some initial values $P^{(0)}, Z^{(0)}, Q^{(0)}$.
  In each iteration obtain a new random sample:
  - Sample $P^{(i)}, Q^{(i)}$ from $\Pr(P, Q \mid X, Z^{(i-1)})$
  - Sample $Z^{(i)}$ from $\Pr(Z \mid X, P^{(i)}, Q^{(i)})$
- For sufficently large m and c mean of sequence

$$Q^{(m)}, Q^{(m+c)}, Q^{(m+2c)}, \ldots$$

converges to $E[Q \mid X]$

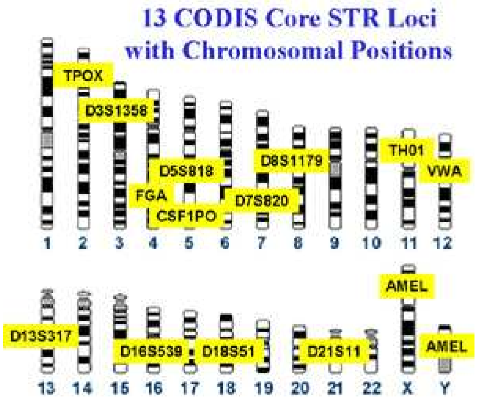
Summary

- **SNPs (single nucleotide polymorphisms)** appear and disappear in populations
- Their frequency influenced by natural selection
- Without recombination, dependency between SNPs on the same chromosome
(**linkage disequilibrium**)
- Recombination creates LD blocks
- LD blocks influence the results of whole-genome association mapping
- Probabilistic models of LD block size, allele frequencies, heterozygocity etc. can reveal population history
- We should consider population structure, which can be estimated using computational methods

Other types of polymorphisms

- **Short indels**

- **Microsatellites a minisatellites**
  (simple short repeating sequences)
  13 locuses as a standard “fingerprint” for comparison of DNA samples in the US courts



- **Transposons** (Alu, LINE, SINE)
  Alu has approx. million copies,
  approx. 1 new copy in 20 newly born

- **Large scale copy number variations**