

Week 4: Comparative genomics

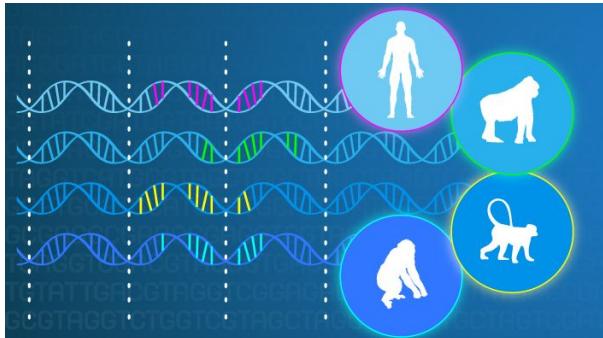
- Whole genome alignment
 - MUMmer & Suffix trees
- Gene/species trees
 - Phylogenetic trees
 - Gene orthology & functional analysis

What is comparative genomics?



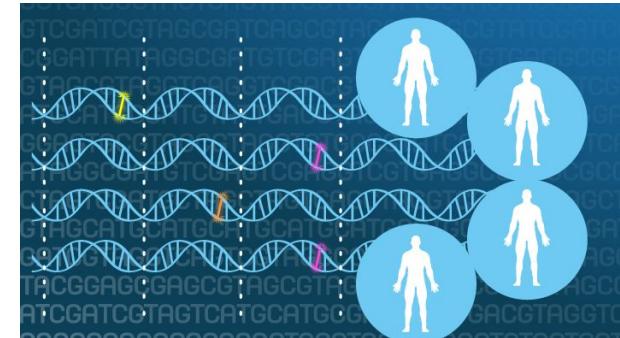
Common features of different organisms

- Conserved DNA.
- Very large distances: gene order & regulatory seqs not conserved.
- Moderate distances: Functional vs. nonfunctional based on negative/purifying selection.



Distinctive features of closely-related species

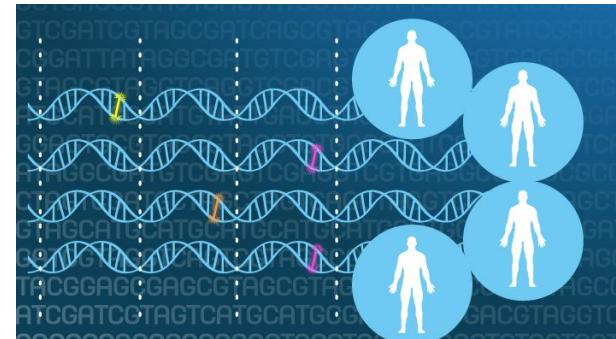
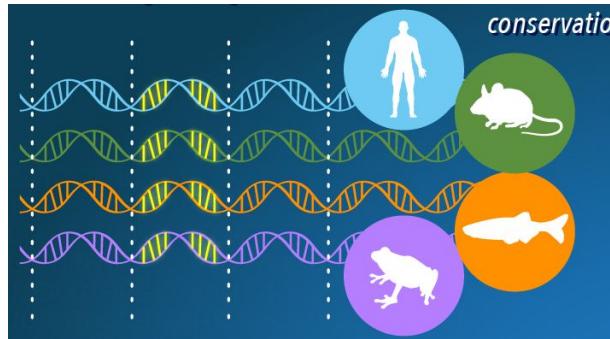
- Unique genomic elements.



Genetic differences within one species

- Genetic variants with a role in a trait/disease.

Why comparative genomics?



- How many times has a feature (anatomy, trait, cellular/molecular property) arisen? been lost?
- How is a disease evolving to avoid immune system?
- What is the sequence of ancestral proteins?
- What are the most similar species?
- What is the rate of speciation?
- Is there a correlation between gain/loss of traits and environment or geographical events?
- Which features are ancestral to a clade, which are derived?
- What structures are homologous, which are analogous?

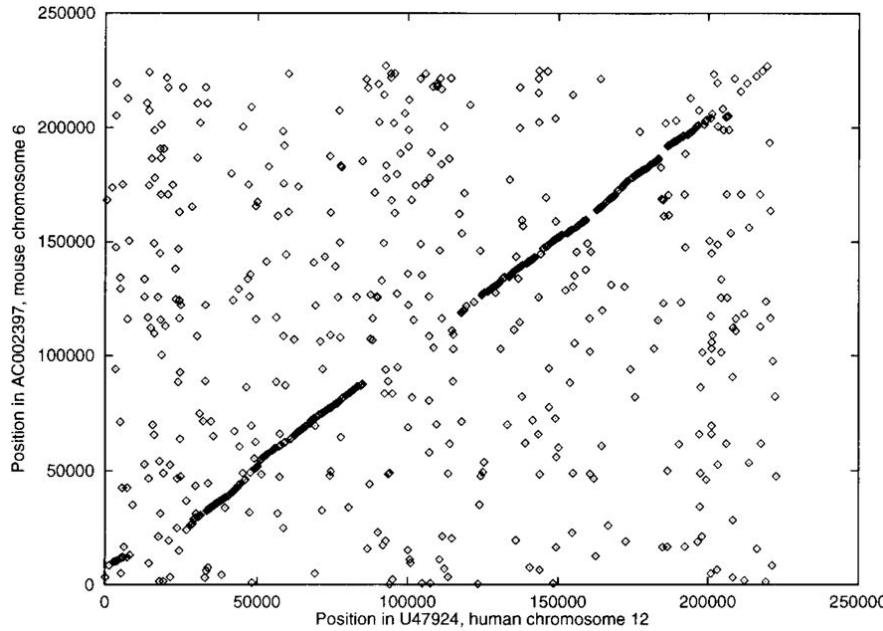
Whole-genome alignment – The problem

Given:

- A pair of large-scale sequences (e.g. chromosomes)
- A method for scoring the alignment (e.g. substitution matrices, insertion/deletion parameters)

Do:

- Construct global alignment: identify all matching positions between the two sequences.



Whole-genome alignment – Need for newer methods

Time & space constraints:

- Traditional DP algorithms require $O(nm)$ time and space (n and m are the lengths of the two sequences).
- As n and m grow to genome scale, too expensive to solve in practice.

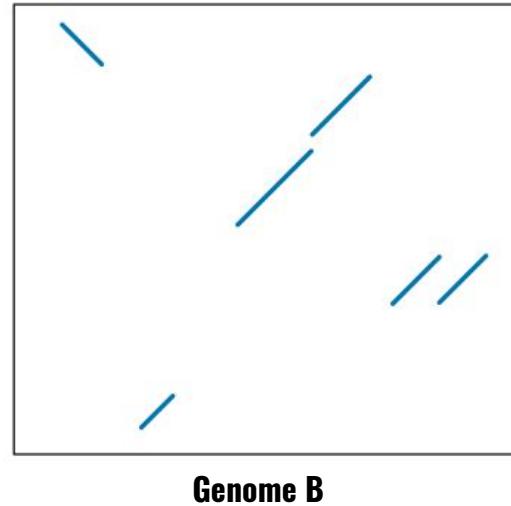
Genomic rearrangements:

- SW & NW → alignments that have fixed order & orientation.
 - Insertions, deletions, and substitutions are the only allowed edit operations (OK for short or well-conserved sequences, like genes).
- Over large evolutionary distances and within a sufficiently large window, genomes almost always contain more complex rearrangements with respect to each other
 - Inversions, transpositions, and duplications all cause breaks in order and orientation that cannot be captured under constant order and orientation.

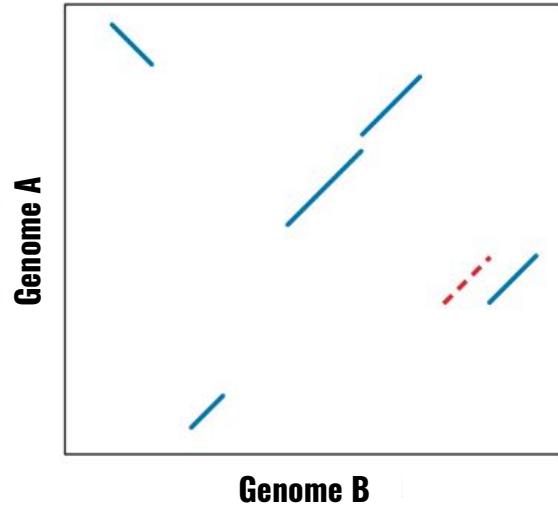
Whole-genome alignment

a

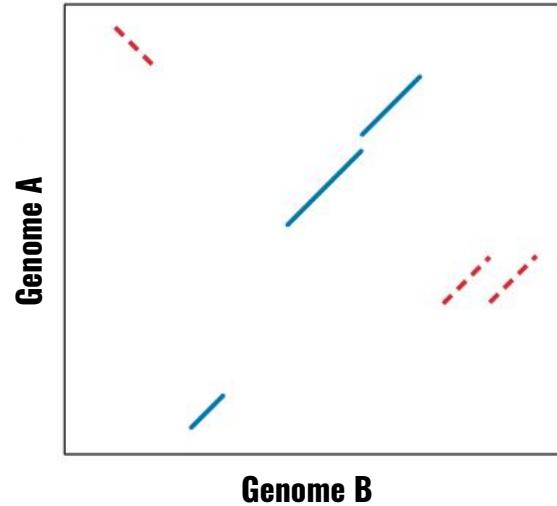
True genome alignment

**b**

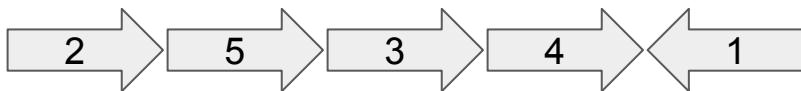
Single-copy

**c**

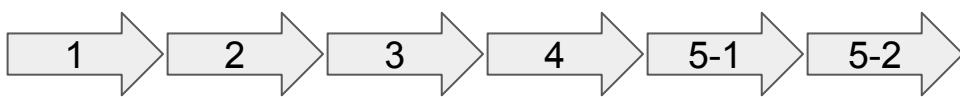
Single order-and-orientation
(global alignment)



Genome A

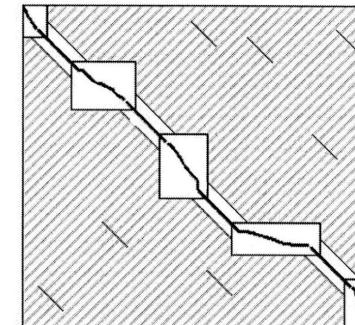
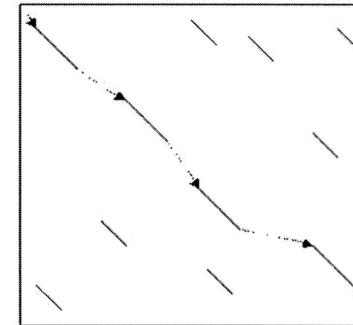
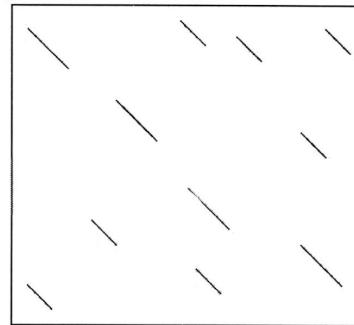


Genome B



What are the alignment for
cases (b) & (c)?

The MUMmer system in 3 steps



Brudno et al. *Genome Research*, 20

General

1. Pattern matching to find seeds for global alignment
2. Find a good chain of anchors
3. Fill in with standard but constrained alignment

MUMmer

1. Suffix trees to obtain MUMs
2. LIS to find colinear MUMs
3. Smith-Waterman and recursive MUMmer for gap filling

The MUMmer system

Delcher et al., *Nucleic Acids Research*, 1999

Given: genomes A and B

1. find all maximal unique matching subsequences (MUMs)
2. extract the longest possible set of matches that occur in the same order in both genomes
3. close the gaps

Step 1: Finding seeds in MUMmer

- *Maximal unique match*:
 - occurs exactly once in both genomes A and B
 - not contained in any longer MUM

Genome A : tcgatcGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAcgactta

Genome B : gcattaGACGATCGCGGCCGTAGATCGAATAACGAGAGAGCATAAtccagag



- Key insight: a significantly long MUM is certain to be part of the global alignment

Suffix trees

- Substring problem:
 - given text S of length m
 - preprocess S in $O(m)$ time
 - such that, given query string Q of length n , find occurrence (if any) of Q in S in $O(n)$ time
- Suffix trees solve this problem and others

Suffixes

$S = \text{"banana\$"}$

suffixes of S

\$ (special character)

a\$

na\$

ana\$

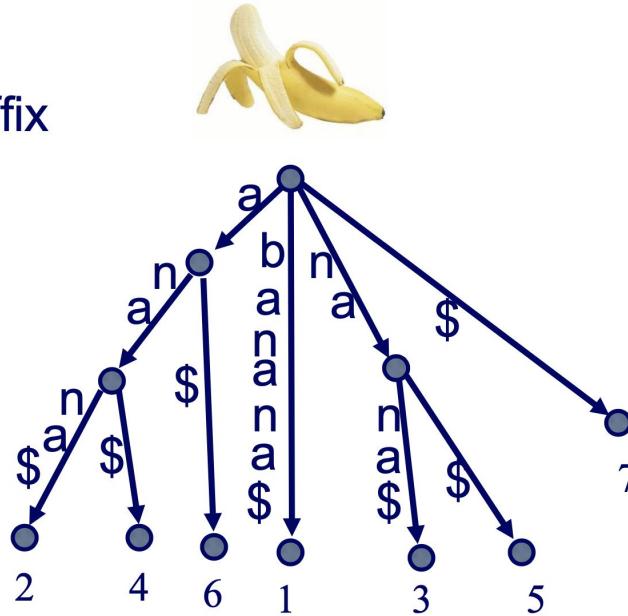
nana\$

anana\$

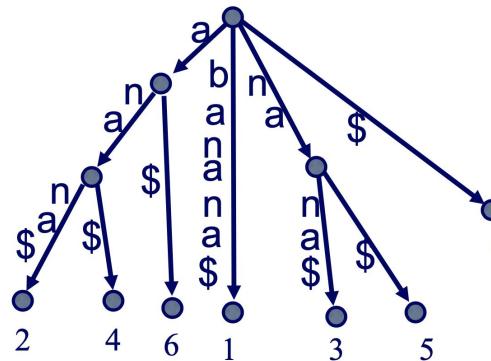
banana\$

Suffix tree – Example

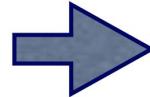
- $S = \text{"banana\$"}$
- Add ‘\$’ to end so that suffix tree exists (no suffix is a prefix of another suffix)



Suffix tree – Definition

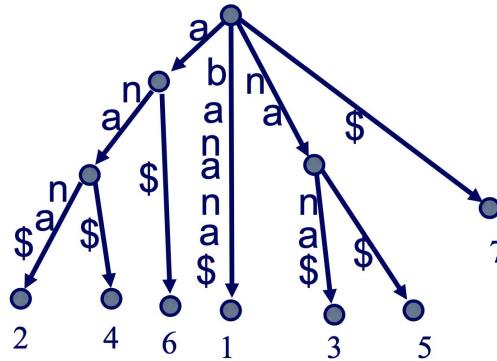


key property



- A suffix tree T for a string S of length m is a tree with the following properties:
 - rooted and directed
 - m leaves, labeled 1 to m
 - each edge labeled by a substring of S
 - concatenation of edge labels on path from root to leaf i is suffix i of S (we will denote this by $S_{i \dots m}$)
 - each internal non-root node has at least two children
 - edges out of a node must begin with different characters

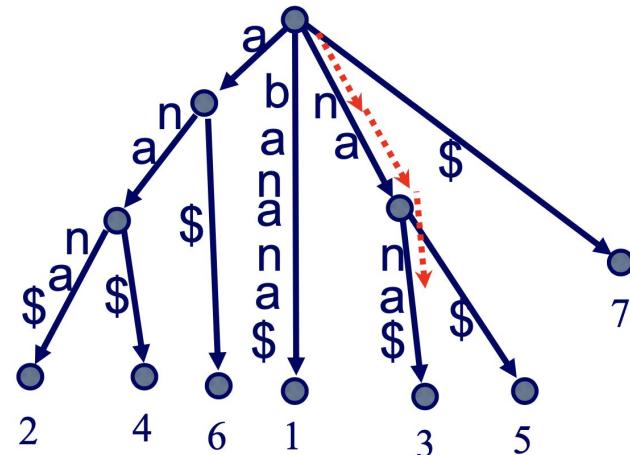
Solving the suffix tree problem



- Assume we have suffix tree T and query string Q
- $\text{FindMatch}(Q, T)$:
 - follow (unique) path down from root of T according to characters in Q
 - if all of Q is found to be a prefix of such a path return label of some leaf below this path
 - else, return no match found

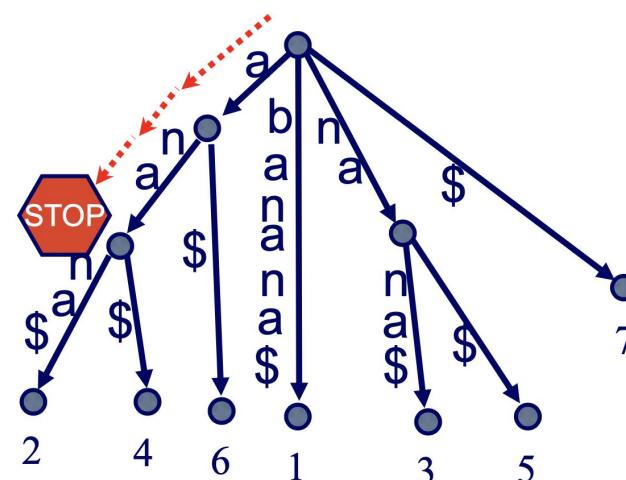
Solving the substring problem

$Q = \text{nan}$



return 3

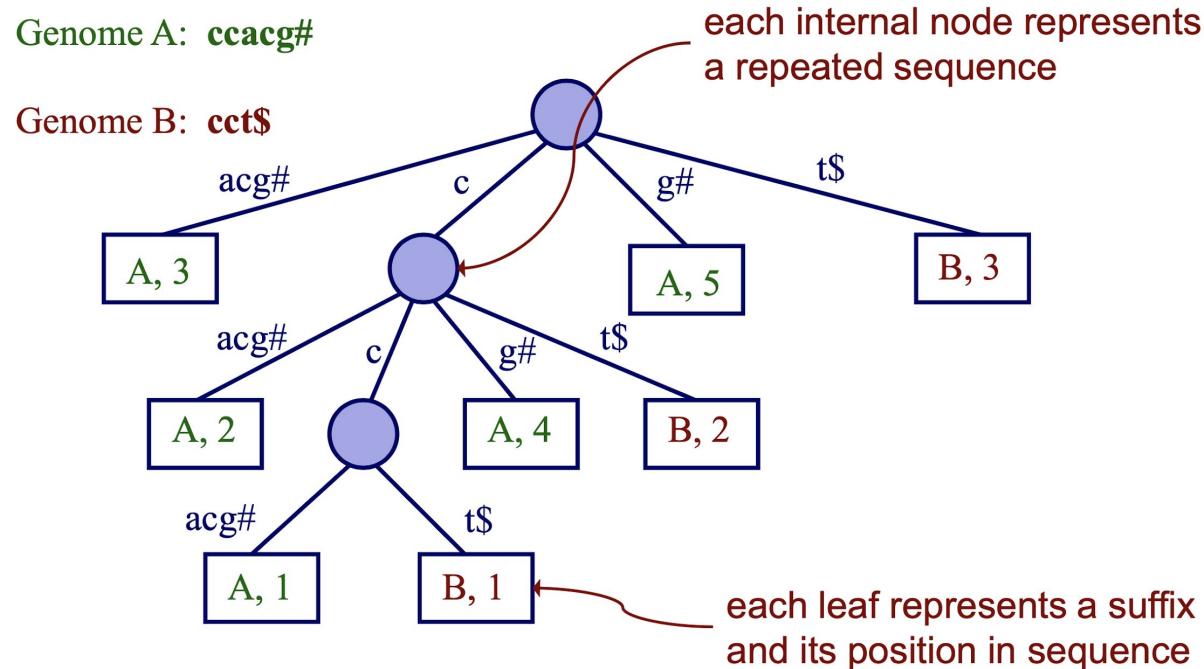
$Q = \text{anab}$



return no match found

MUMs and Generalized suffix trees

- Build one suffix tree for both genomes A and B
- Label each leaf node with genome it represents

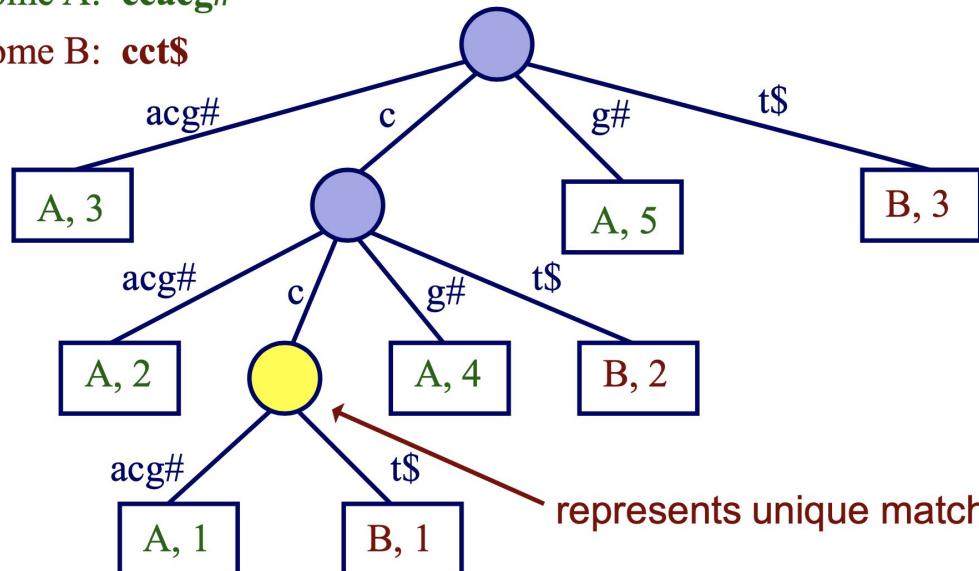


MUMs and Suffix trees

- Unique match: internal node with 2 children, leaf nodes from different genomes
- But these matches are not necessarily maximal

Genome A: ccacg#

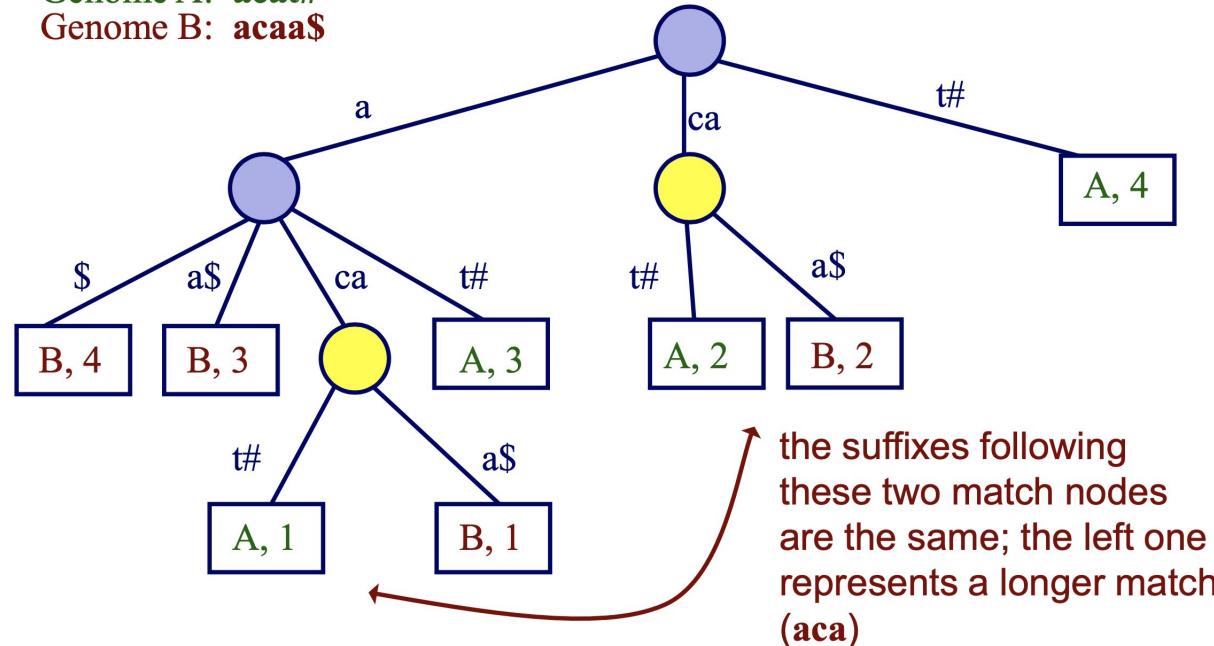
Genome B: cct\$



MUMs and Suffix trees

- To identify maximal matches, can compare suffixes following unique match nodes

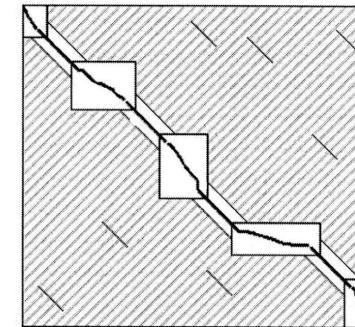
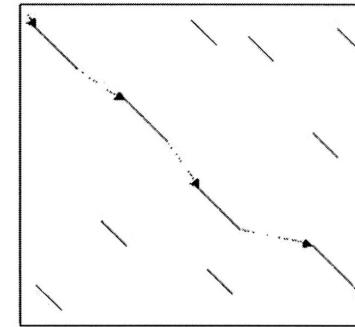
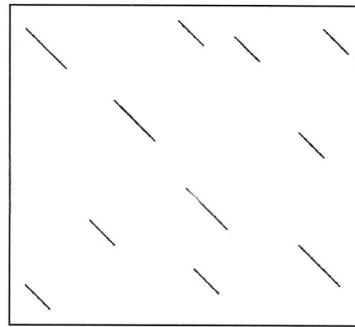
Genome A: acat#
Genome B: acaa\$



Using suffix trees to find MUMs

- $O(n)$ time to construct suffix tree for both sequences (of lengths $\leq n$)
- $O(n)$ time to find MUMs - one scan of the tree (which is $O(n)$ in size)
- $O(n)$ possible MUMs in contrast to $O(n^2)$ possible exact matches
- Main parameter of approach: length of shortest MUM that should be identified (20 – 50 bases)

The MUMmer system in 3 steps



Brudno et al. *Genome Research*, 20

General

1. Pattern matching to find seeds for global alignment
2. Find a good chain of anchors
3. Fill in with standard but constrained alignment

MUMmer

1. Suffix trees to obtain MUMs
2. LIS to find colinear MUMs
3. Smith-Waterman and recursive MUMmer for gap filling

Step 2: Chaining in MUMmer

- Sort MUMs according to position in genome A
- Solve variation of *Longest Increasing Subsequence* (LIS) problem to find sequences in ascending order in both genomes

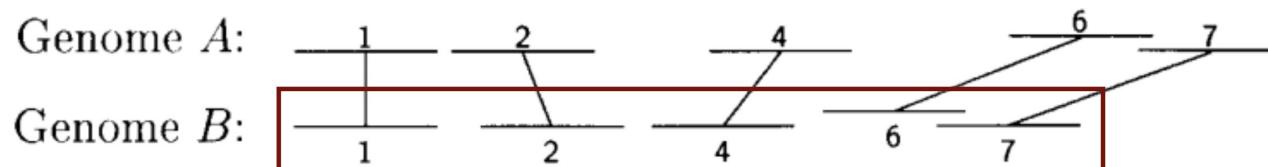
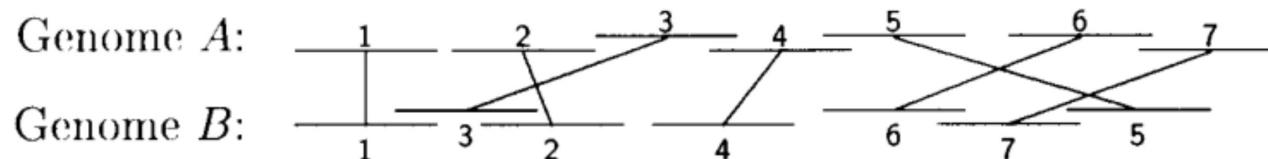
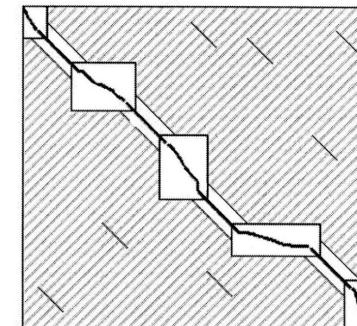
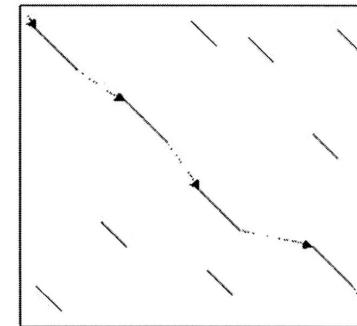
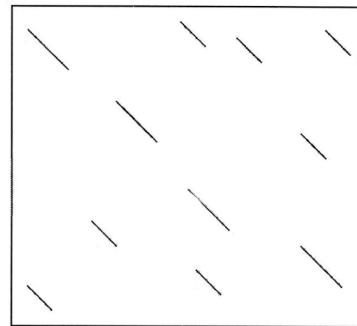


Figure from: Delcher et al., *Nucleic Acids Research* 27, 1999

Finding longest subsequence

- Unlike ordinary LIS problems, MUMmer takes into account
 - lengths of sequences represented by MUMs
 - overlaps
- Requires $O(k \log k)$ time where k is number of MUMs

The MUMmer system in 3 steps



Brudno et al. *Genome Research*, 20

General

1. Pattern matching to find seeds for global alignment
2. Find a good chain of anchors
3. Fill in with standard but constrained alignment

MUMmer

1. Suffix trees to obtain MUMs
2. LIS to find colinear MUMs
3. Smith-Waterman and recursive MUMmer for gap filling

Types of gaps in a MUMmer alignment

1. SNP: exactly one base (indicated by $\hat{}$) differs between the two sequences. It is surrounded by exact-match sequence.

Genome A: cgtcatggcg \hat{t} tcgtcgttg
Genome B: cgtcatggcattcgtcgttg

2. Insertion: a sequence that occurs in one genome but not the other.

Genome A: cgggtaaccgc.....cctggcggg
Genome B: cgggtaaccgcgttgc \hat{t} ccggtaaccgc \hat{t} ccgg

3. Highly polymorphic region: many mutations in a short region.

Genome A: ccgcctcgcc \hat{t} gg.gctggcgcccgtc
Genome B: ccgcctcgccagttgaccgc \hat{t} ccgcgtc

4. Repeat sequence: the repeat is shown in uppercase. Note that the first copy of the repeat in Genome B is imperfect, containing one mismatch to the other three identical copies.

Genome A: cTGGGTGGGACAACGTaaaaaaaaaaaTGGGTGGGACAACGTc
Genome B: aTGGGTGGGGC \hat{g} ACGTggggggggggTGGGTGGGACAACGTa

Figure from: Delcher et al., *Nucleic Acids Research* 27, 1999

Step 3: Close the gaps

- SNPs:
 - between MUMs: trivial to detect
 - otherwise: handle like repeats
- Insertions
 - simple insertions: trivial to detect
 - transpositions (subsequences that were deleted from one location and inserted elsewhere): look for out-of-sequence MUMs

Step 3: Close the gaps

- Polymorphic regions
 - short ones: align them with dynamic programming method
 - long ones: call MUMmer recursively with reduced minimum MUM length
- Repeats
 - detected by overlapping MUMs

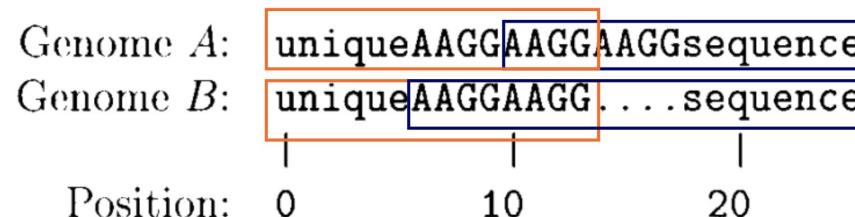
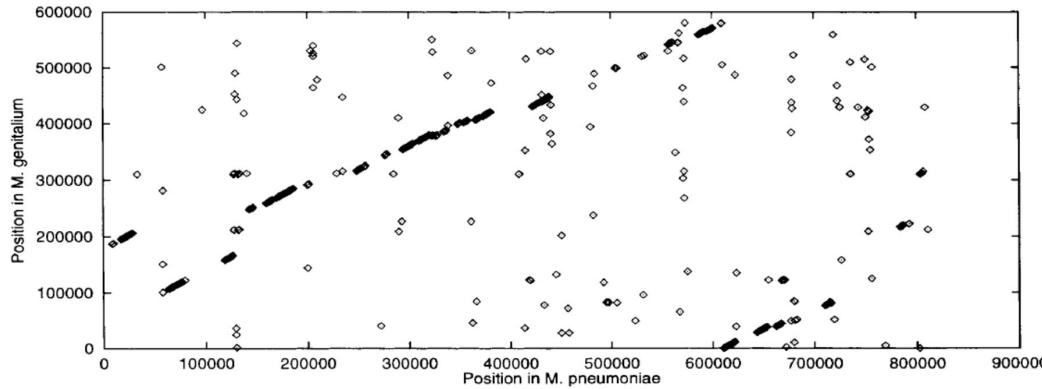


Figure from: Delcher et al. Nucleic Acids Research 27, 1999

MUMmer performance

FASTA on
1000 base
pair segments



MUMmer

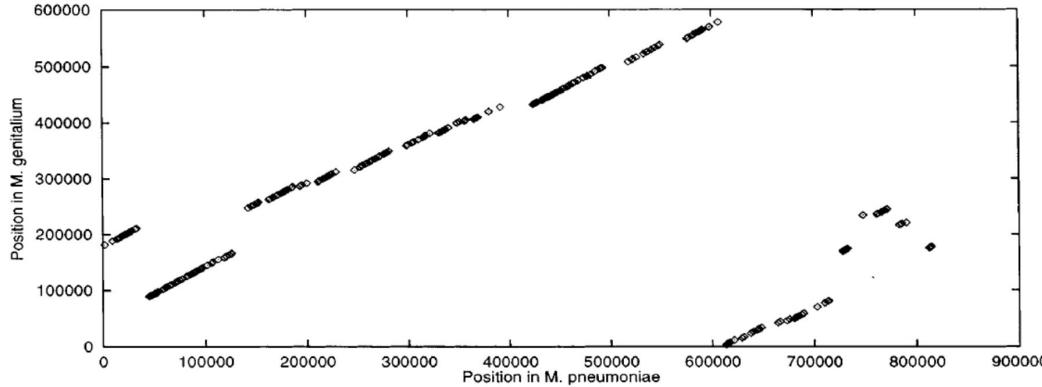


Figure from: Delcher et al. Nucleic Acids Research 27, 1999

MUMmer performance

- *Mycoplasma* test case
- Suffix tree: 6.5s
- LIS: 0.02s
- Smith-Waterman: 116s

- FASTA baseline: many hours

DEC Alpha 4100



[Centre for Computing History](#)

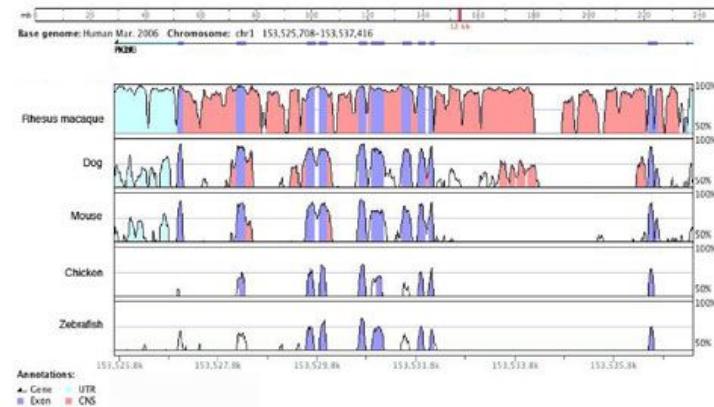
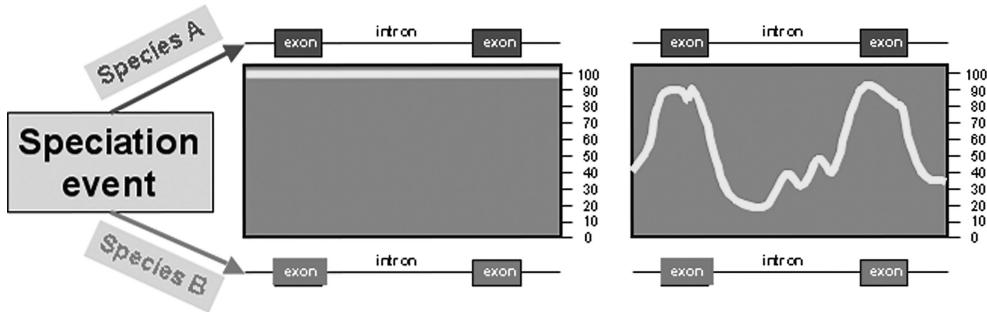
Limitations of MUMmer

- MUMs are perfect matches, typically \geq 20-50 base pairs
- Evolutionarily distant may not have sufficient MUMs to anchor global alignment
- How can we tolerate minor variation in the seeds?

Paper report & discussion: Longevity of MUMmer

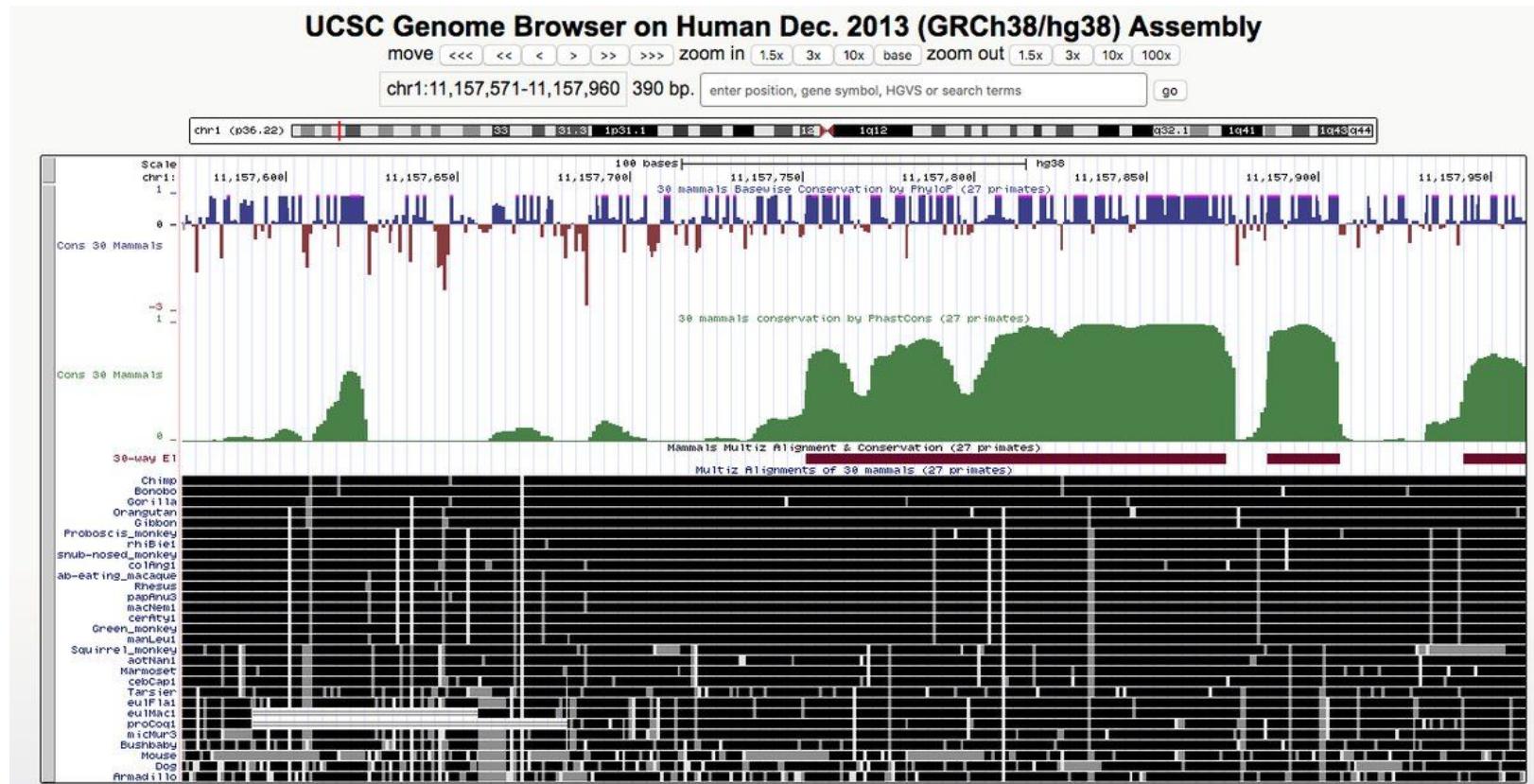
	Biological / conceptual	Technological / experimental	Algorithmic / methodological	Hardware / software
MUMer v1				
MUMer v2				
MUMer v3				
MUMer v4				

Comparative genomics

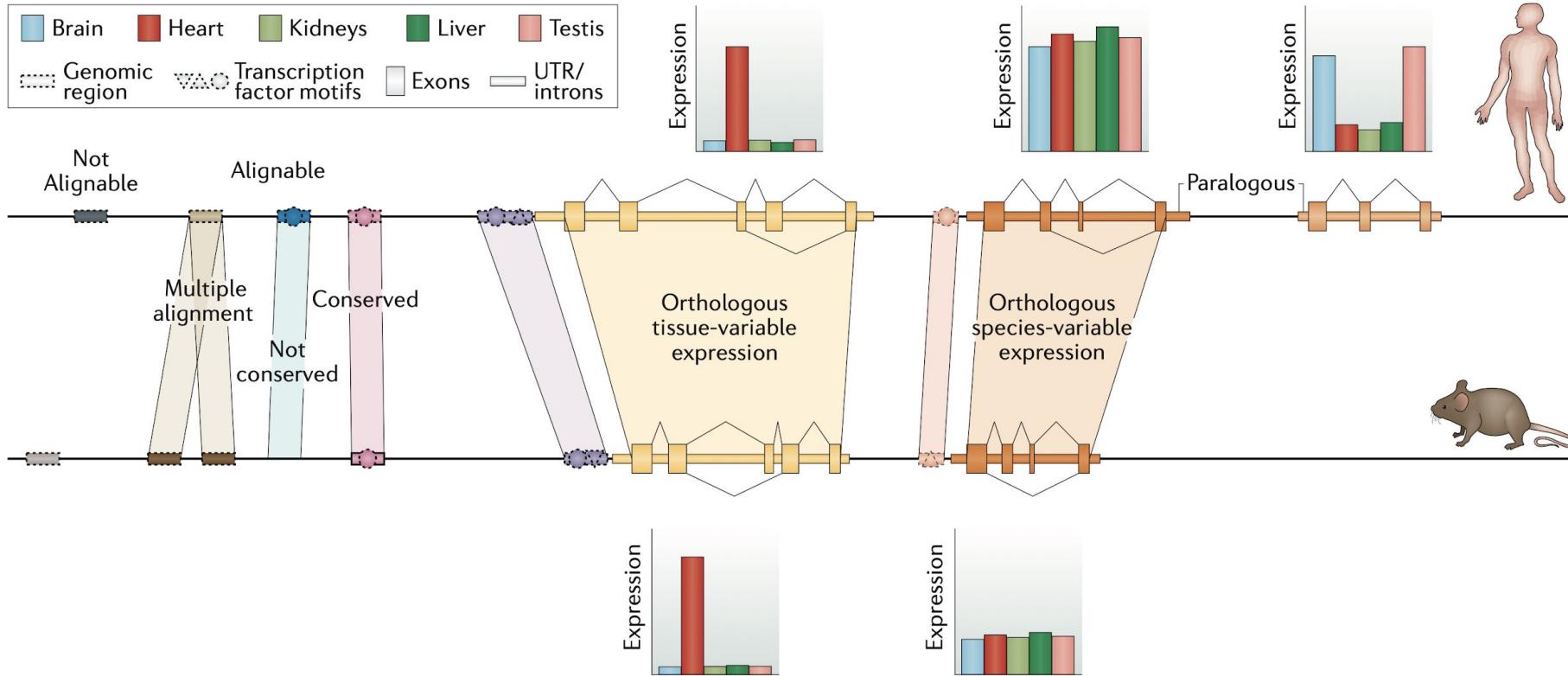


Comparative genomics

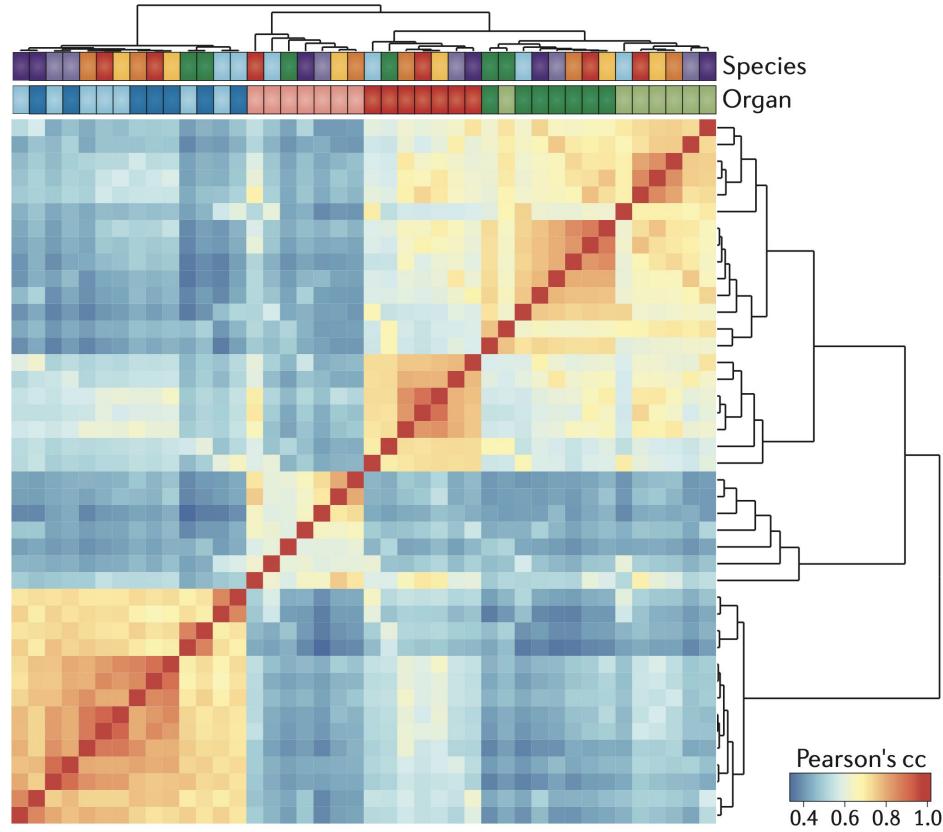
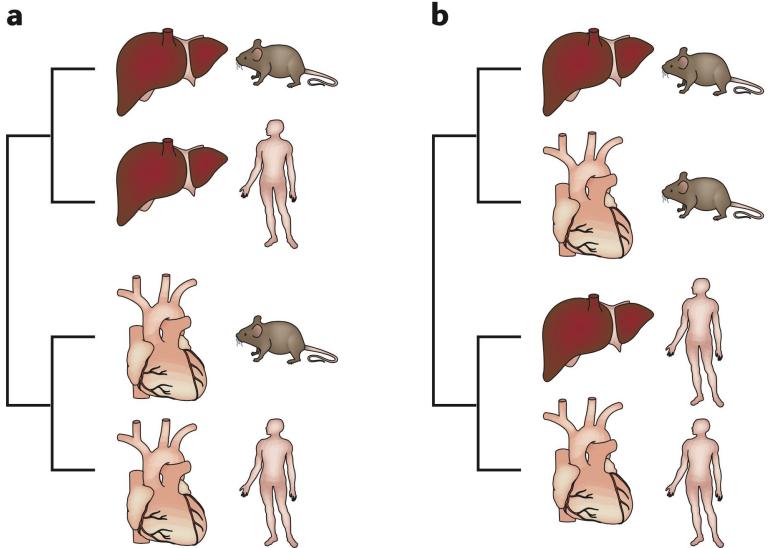
Multiple alignments and measurements of evolutionary conservation for 30 species (27 primates)



Comparative transcriptomics



Comparative transcriptomics



Comparative -omics

How conservation can be used to identify the most likely functional variants in a disease locus?

