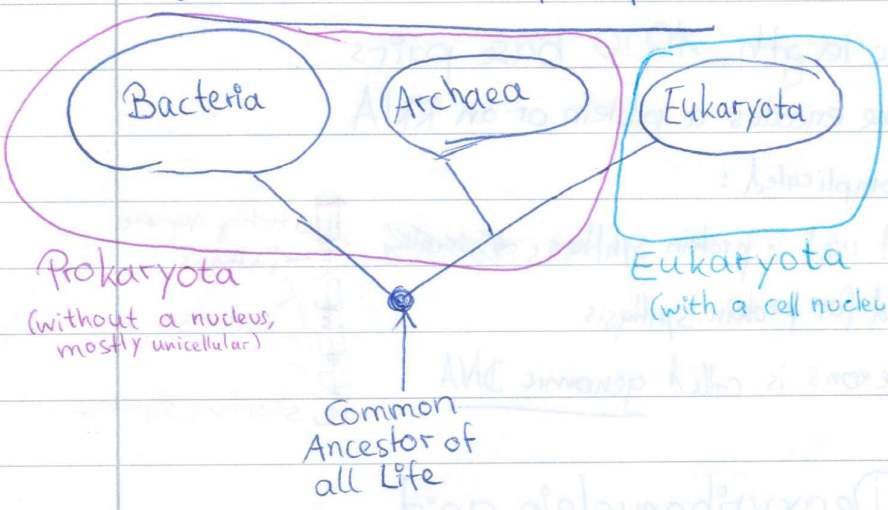
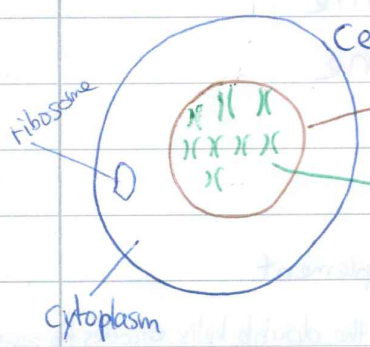
Biological Background Knowledge

3 domains of life



Phylogenetic tree based on the 16S rRNA gene

Eukaryota, human example



→ larger genome ≠> more complex organism, often due to redundant information (see plants, marbled lungfish)

human genome: 23 pairs of chromosomes,
~3 billion basepairs ×2
98% non-coding DNA
→ only 2% of the genome used to build proteins

haploid: each chromosome once, → sexual reproduction
diploid: two versions of each chromosome, one from each parent

Chromosomes: long DNA molecules

- Prokaryotes: one chromosome
- Eukaryotes: many chromosomes, organized in pairs (paternal/maternal)
- homologous: paired chromosomes

spaces between genes
↓
"non-coding"

500–4.000 genes per chromosome

2136

Genes: - coding parts of DNA, each gene encodes for either RNA or a protein

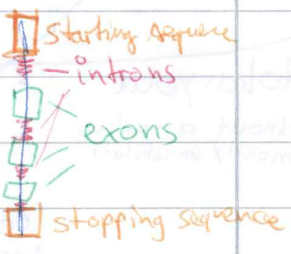
- average gene length: 1000 base pairs

Prokaryotes: A gene encodes a protein or an RNA

Eukaryotes: More complicated:

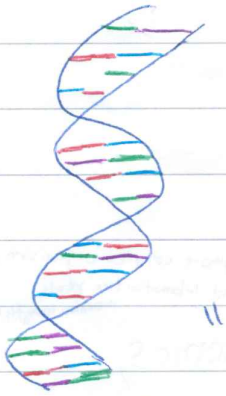
- introns: not used in protein synthesis ~~"non-coding"~~
- exons: used for protein synthesis

- DNA data in exons is called genomic DNA



DNA: ~~Desoxy~~ Deoxyribonucleic acid

- double helix structure, organized in chromosomes



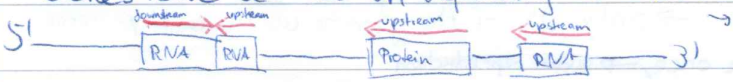
4 nucleic ~~acids~~ acids/base pairs/nucleotides:

adenine – thyamine

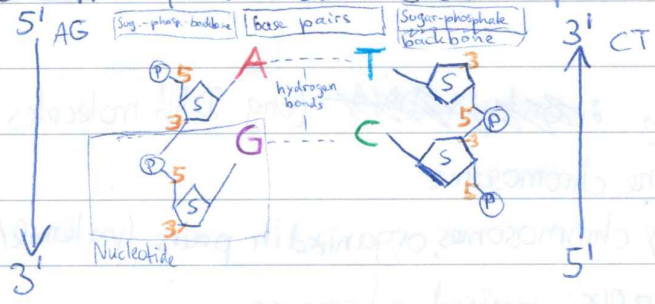
cytosine – guanine

"blueprint of life"

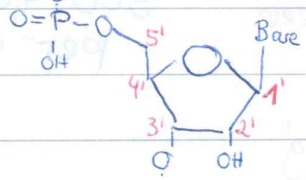
- DNA strand: Single string of DNA without the complement
- Genes have a direction depending on which strand of the double helix encodes the gene → must be read from the correct side to be recognized



~~reading~~ convention: from 5' end to 3' end ("5 prime to 3 prime")



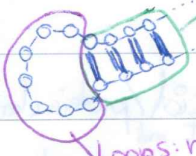
Labelling of carbon atoms in the nucleotide sugar-ring:



Biological Background Knowledge II

RNA: Ribonucleic acid

- single-stranded sequence
- Thyamine is replaced by Uracil
- RNA is a copy of a coding DNA strand (a gene)
  - involved in Transcription to construct either:
    - coding RNA: ~~used~~ also involved in ~~Transcription~~ translation (protein synthesis)
    - non-coding RNA: some other direct function in the cell
- has a secondary structure (and even a tertiary): ↪ influence on its function



Stem: complementary bases bind A↔U, C↔G

Loops: no matching bases

matching bases cannot mutate independently from each other
↪ importance for evolution of RNA

- types of RNA

~3% coding RNA:

mRNA (messenger RNA)
transports RNA data to the ribosome for protein synthesis

non-coding RNA

rRNA (ribosomal RNA)
carries out the translation in the ribosome via catalysis
↪ one of a few genes that are common to all species
↪ most well-known: 16S rRNA gene
↪ can be used to infer evolutionary relationships amongst all species, useful for ~~cheap~~ classification of bacteria

tRNA (transfer RNA)
brings in the amino acids

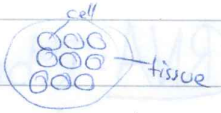
...

- Transcriptome: set of all RNA molecules in a cell

↪ reflects the activity in a cell → temporal and spatial component
- specialized cells → different genes are active
- point of time changes transcriptome

- Meta-Genome: blind sequencing of all genetic material of a bacterial community → many species, figure out microbial diversity. Can be done at whole-genome level → metagenomics
  ↪ single gene level → metagenetics, e.g. 16S RNA for Bacteria

Proteins: do lots of stuff in the body

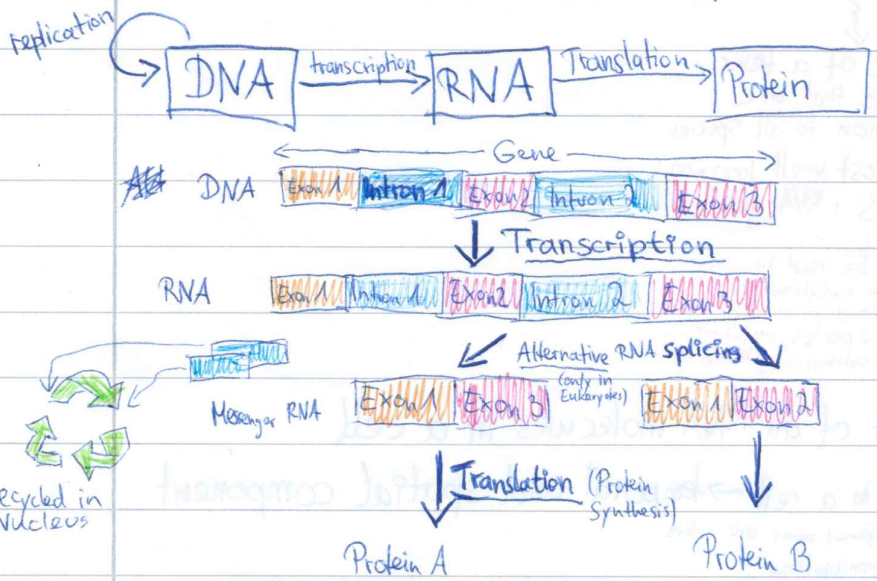


- Structural proteins: tissue building blocks
- Enzymatic proteins: catalysts of biochemical reactions in the body
  - examples: oxygen transport, immune defense, provide & store energy, ...
- human: ~20.000 proteins
- Sequence/string of amino acid characters, protein letters are called residues
- have primary, secondary, tertiary and quarternary structure

Amino acids: building blocks of proteins, 20

- 3 DNA/RNA characters encode for one amino acid/protein character
  - such a triplet is called a Codon
  - extra START and STOP codons, STOP codon is not an AA
- Protein alphabet / AA alphabet:
  - DNA → Protein: not ambiguous, but redundant
  - Protein → DNA: ambiguous, several DNA triplets encode for the same AA
- Synonymous substitutions/mutations (GCC → GCT = Alanine → Alanine)
- non-synonymous substitutions/mutations (GGT → GTT = Glycine → Valine)
- 3rd codon position is less vulnerable to mutations
  - ↳ mostly results in the same AA after mutation

Central Dogma of Molecular Biology



Transcription happens inside the nucleus, Translation happens outside (at the ribosome)

Biological Background Knowledge III

DNA sequencing: reading the nucleotide bases in a DNA molecule

- Extended DNA ~~sequence~~ alphabet

  Since sequencing methods are not ~~exact~~ exact

  → character set to denote "A", "A or C", "A or C or G", "A or C or G or T", ... and a gap-character (needed for alignment)

- **Sanger Sequencing** (first generation sequencing, ≈1977)

  ↳ Chain Termination Method

  - Informal game analogy:

    ~~You think of a~~ ① You think of a 200 digit number, others must guess it

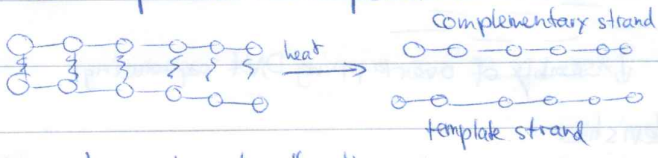
    ② You ~~start~~ read the number in your head

    ③ Someone randomly interrupts

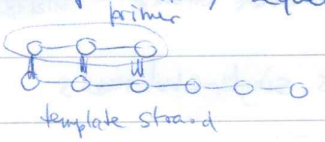
    ④ You answer current position and digit.

    ⑤ start again with step 2.

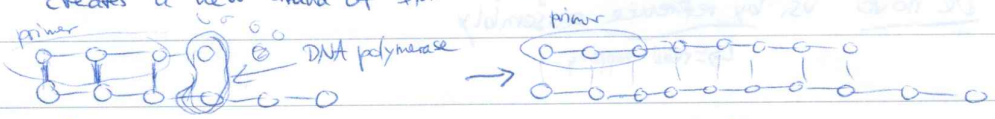
- ① heat up DNA to separate its strands



- ② lower temperature to allow the primer sequence to bind to its complementary sequence in the template DNA



- ③ raise temperature s.t. the enzyme polymerase 3 combines to the DNA and creates a new strand of DNA



- ④ dNTPs are added by the enzyme until a ddNTP is added which stops synthetisis

  ↳ Creates DNA strands of different length

  ddNTP: colored & chemically altered version of A, C, G, T

- ⑤ apply heat again to separate
- ⑥ let the pieces run a race on ~~a~~ a gel with 4 lanes, (gel eletrophoresis)

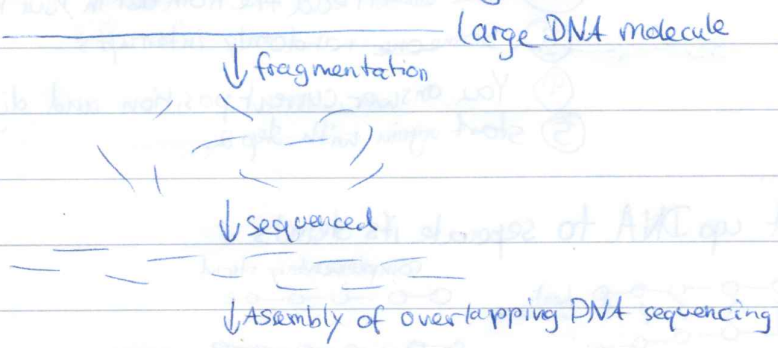
  Smaller pieces are faster than longer ones, letter can be seen by color

Sanger Sequencing	Next Generation Sequencing (pyrosequencing)
~~- needs radioactive bacteria~~	- cheap
- expensive	- fast, many sequences
- slow, few sequences	- short sequences
- long sequences	- less accurate
- high accuracy	

pyrosequencing: Die Dinger geben direkt Lichtab, was gemessen werden kann → kein Abbruch

Shotgun Sequencing: reading the genome

1. Break up genome randomly into fragments
2. Read fragments
3. Assemble fragments into genome



Important characteristics

- Coverage (how many ~~fragments/reads~~ cover one ~~nucleotide~~ on the genome

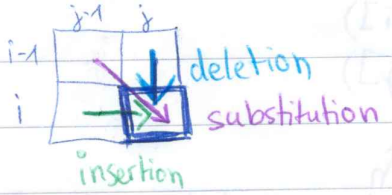
- Fragment length
- Paired-end (ATTG-----GTC) vs. single-end reads
  - ↳ additional information, makes assembly process easier
  - we know the length of this gaps, same for all fragments
- De novo vs. by reference assembly
  - ↳ = read mapping

Pairwise Sequence Alignment

- Sequence Alignment: compare strings to infer their similarity
- online sequence alignment: no preprocessing for index building
- Edit distance (≥ 0, $=0$ iff same string, symmetric, Δ-inequality)
  - → Damerau-Levenshtein: $sub(a,b) = sub(b,a) = del(a) = ins(b) = 1$, $a \neq b$
  - → Hamming distance: ~~$del(a) = ins(a) = \infty$~~, count only substitutions
  - → minimum cost of a set of edit operations to transform a into b
- alignment: for example $\begin{pmatrix} ACG--A \\ ATGCTA \end{pmatrix}$ → cost 3
  no ~~gap~~ (–) allowed
  here: $x = ACGA$, $y = ATGCTA$

Edit distance via dynamic programming

$$a[0,0] = 0$$
$$a[i,0] = a[i-1,0] + del(x[i])$$
$$a[0,j] = a[0,j-1] + ins(x[j])$$
$$a[i,j] = \min \begin{cases} a[i-1,j-1] + sub(x[i], y[j]) \\ a[i-1,j] + del(x[i]) \\ a[i,j-1] + ins(y[j]) \end{cases}$$



$O(m \cdot n)$

Needleman-Wunsch: Global alignment

- maximize similarity, $sub(a,a) > 0$, $ins(a), del(a), sub(a,b) < 0$
  - indel

Substitution Matrices: BLOSUM

(BLOcks of Amino Acid Substitution Matrix)
- → for sequence alignment of protein sequences
- → based on observed (local) alignments
- → got data from conserved regions of protein families w.o. gaps
- → different matrices for closely / distant related sequences
- → $sub(a,b) = \log\left(\frac{P \text{ biological sense}}{P \text{ by chance}}\right)$ ← log-odds score
  - → positive score: more likely

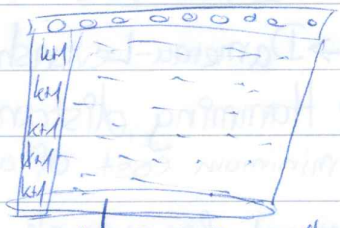
8136

Hamming distance

Input: String x, String y, integer k

Wanted: Alignments with hamming distance $\leq k$ (x–y–)

$$a[i,j] = \begin{cases} k+1, & j=0,\ i>0 \\ 0, & i=0 \\ a[i-1,j-1], & x[i]=y[j] \\ a[i-1,j-1]+1, & x[i] \neq y[j] \end{cases}$$



traceback from all values in the last row that are $\leq k$ to a zero.

Smith-Waterman: Local Alignment

- Needleman-Wunsch with free taxi drives

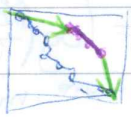
$$a[i,j] = \begin{cases} 0, & i=0 \vee j=0 \\ \max \begin{cases} 0 \rightarrow \text{free taxi drive} \\ a[i-1,j-1] + sub(x[i], y[j]) \\ a[i-1,j] + del(x[i]) \\ a[i,j-1] + ins(y[j]) \end{cases} \end{cases}$$

differences to Needleman-Wunsch
- other initialization (free drive to your starting position)
- best value stands anywhere in the table (free drive to your ending position)

Global vs. Local ~~Alignment~~ Alignment

- find the highest scoring alignment between two strings
- good for similar strings with similar size

- find the highest scoring alignment between two substrings
  - ↳ global alignment in a subrectangle of the DP-matrix
- idea: free taxi rides



- better for different strings with small similar regions (i.e. genes in different species)
  - ↳ expected to have small regions of similarity

Searching Similar Sequences

- Assumption: Similar sequences are evolutionary and/or functionally related

- Naive approach: Pair-wise alignment of query sequence with every sequence in the database via Smith-Waterman, report best matches
  → $O(m \cdot n \cdot d)$ → too slow, since database size d often large

- BLAST: Basic Local Alignment Search Tool
  - ↳ heuristic to find similar sequences, widely-used
  - ↳ good for large, fast evolving database; small number of queries (compared to DB size); interested in distantly related sequences → example: found new gene in mouse, perform BLAST search on human genome to check if humans have a similar gene
  - ↳ different flavors: (database, query) $\in$ {nucleotide sequence, protein sequence, nucleotides translated into protein sequences}^{2}
  - ↳ Idea: reduce search space by ignoring dissimilar regions

  ① Seeding: find common subwords between query and database sequences → seeds
  - collect all substrings of length k of the query sequence
  - for each of this substrings, build neighborhood with "similar" subwords of length k
    - ↳ defined via substitution matrix, BLOSUM for proteins, +2/-3 (or +5/-4) for DNA match/unismatch,

    add only subwords with a similarity $\geq$ threshold T.
    → neighbors are only used to find more positions to look at

  ~~② Search: Scan~~

  ② Scanning: ~~Scan the database for exact matches of sub~~ seeds

  ③ ② Extension: Starting from seeds, extend alignment in both directions → High-scoring Segment pairs (HSP)
  - ↳ try to extend the alignment to the left and to the right from the seed, stop if current total score < CurrentMaxScore − X, trim back the alignment to curr MaxScore

③ Evaluation: Assess the statistical significance of each HSP

↳ are the sequences biologically related or just by chance similar?

↳ Smith-Waterman local alignment scores between two random sequences follow Gumbel extreme value distribution, $p(S \geq x) = 1 - \exp(e^{-\lambda(x-\mu)})$, depending on substitution matrix, gap penalties, sequence length, nucleotide frequencies

↳ BLAST reports e-value (expectation value): expected number of times that an unrelated database sequence would obtain a score S higher than x by chance

$E \approx 1 - e^{-p(S>x) \cdot d}$

Genome assembly: ~~reconstructing~~

taking a large number of short DNA sequences and putting them back together, reconstruct genome

Genome (multiple copies) ← A long DNA molecule

ACTTGCTA...

↓ Shotgun sequencing

Short reads

↓

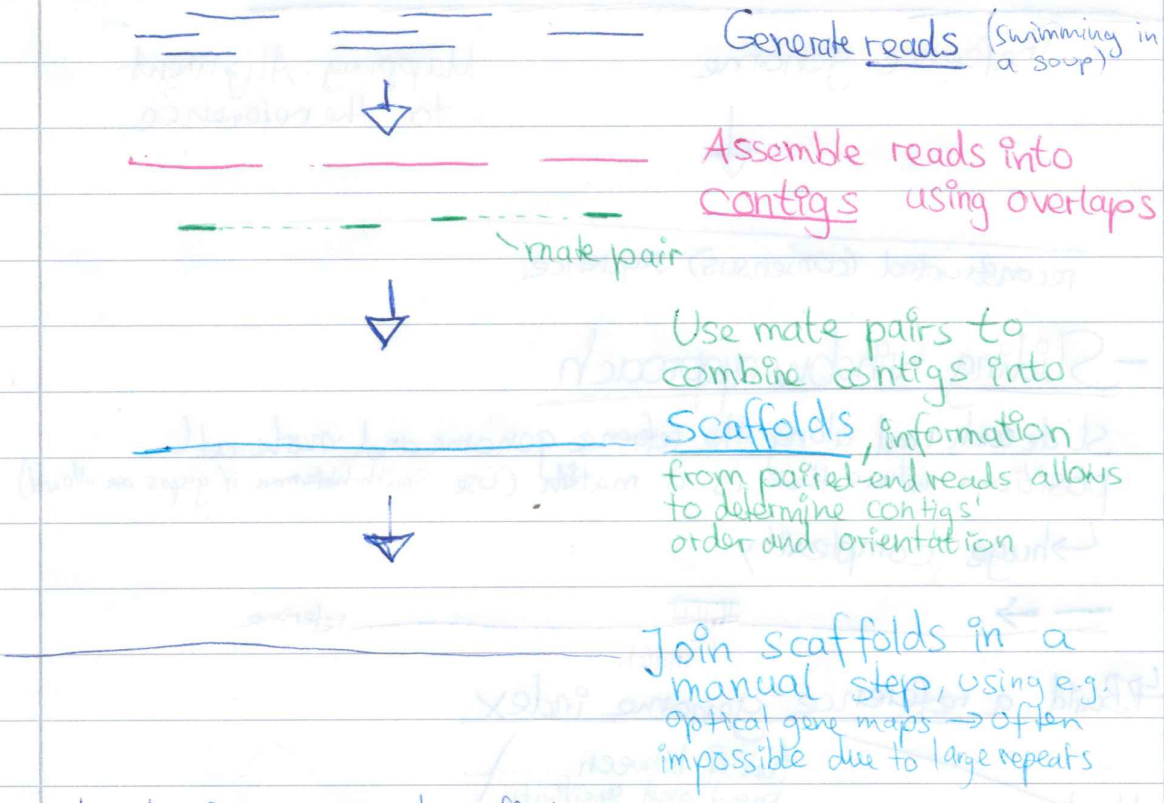
Assembler

↓

~~Reconstructed~~ (consensus) sequence

De novo assembly

- exploit read overlaps to build a (novel) genome sequence from scrath



- length of contigs and scaffolds is an important metric of assembly quality
- 2 approaches for de novo assembly/reconstructing sequence:

Overlap graphs

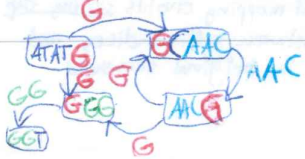
nodes: each read

edges: overlaps between reads, found by pair-wise alignments

wanted: hamiltonian path (visit each node exactly once)

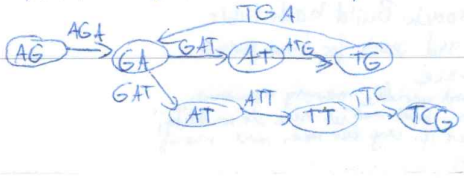
- no efficient algorithm for finding hamiltonian path known, quadratic time complexity of pair-wise alignments

→ good if small number of reads with significant overlap, e.g. Sanger sequencing



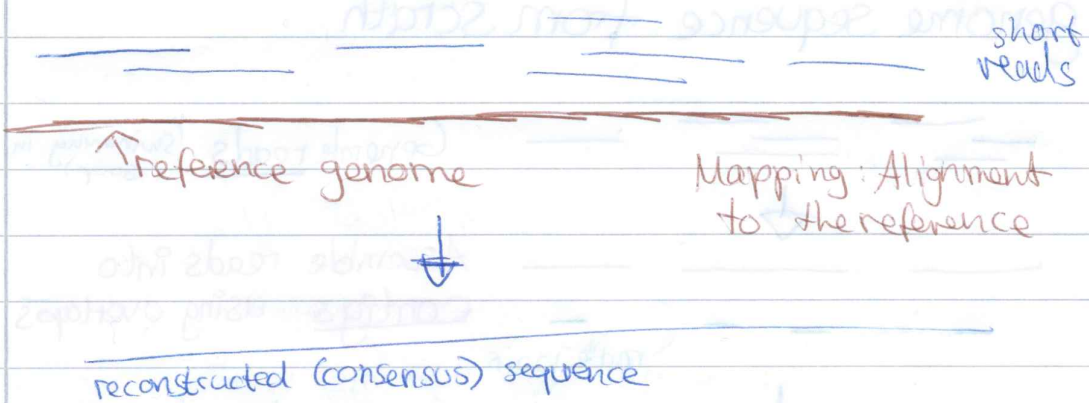
de Bruijn graphs

- ~~edges~~ each read is decomposed into k-mers
- each unique k-mer is represented by an edge in the graph
- nodes are (k-1)-mers: prefix + suffix of the k-mer associated with the edge connecting them
- wanted: eulerian path (visit each edge exactly once)
- efficient algorithm for eulerian path exists
- compact representation of repeats (duplicate k-mers within single node, longer repeats form a single series of adjacent nodes)
- Graph building time is linear in number of reads
- ~~the~~ sequencing errors require tip removal and bubble removal
- used in most modern NGS sequencers



By-reference assembly

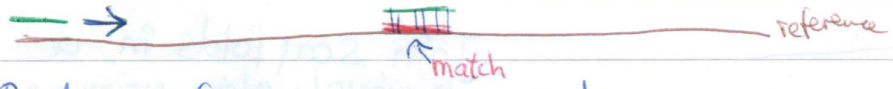
- map reads to the known reference sequence - usually genome of the same species or a close relative / close relatives
- faster and less memory intensive than de novo



- Sliding window approach

slide each read along the reference genome and mark all positions where there is a match (use Smith-Waterman if gaps are allowed)

↳ huge complexity



↳ Build a reference genome index

trade-off between speed and sensitivity → heuristics using the index

Hashing-based

use hash table to store positions of all k-mers in a genome, $k \ll$ read size
→ linked list of positions for each k-mer

0. for each read, select one "proxy" k-mer (leftmost or middle part since better read quality)
1. find all positions of this k-mer in the genome → seeds
2. for each seed, try to extend the alignment (map the rest of the read), with mismatches/gaps like in Smith-Waterman

Variants & Optimizations:

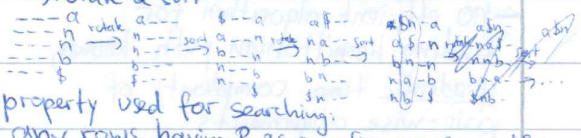
- inexact matches with spaced seeds
  - ↳ binary mask defines positions where mismatches are allowed
    - ↳ allows seeds with mismatches
- Multiple k-mers per read
  - ↳ require at least n seed matches for a mapping location to be considered
- inverted approach: Build hash table from reads and search for k-mers present in the reference
  - ↳ smaller and variable memory requirements, depending on input read set size and variability; may be slower if very few reads since scanning whole genome

Burrows-Wheeler transform (BWT)-based

- better in both time and memory usage, used for data compression

1. Write down all cyclic rotations of the source string S ← has end-of-line marker (counts as smallest)
2. Sort the rows lexicographically
3. Store the last column → BWT(S)

- rows of the matrix form a sorted list of suffixes → allows efficient search for substring occurrences
- BWT is reversible → everything can be reconstructed from the last column
  - ↳ trick: in order to obtain first column, sort last one, in order to obtain second column, sort (last one + first one),
    - ↳ rotate & sort



- property used for searching: any rows having P as a prefix are consecutive
- use FM-index to make matrix traversal efficient
  - ↳ last-to-first mapping avoids sorting step
  - ↳ mapping between suffix indices and positions in original sequence

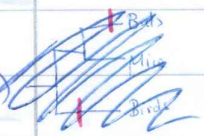
Multiple Sequence Alignment

Why? : compare more than two sequences at a time
↳ input for phylogenetic reconstruction
↳ discover important (conserved) parts of a protein family
↳ group of evolutionary related genes/proteins in different species with similar function/structure

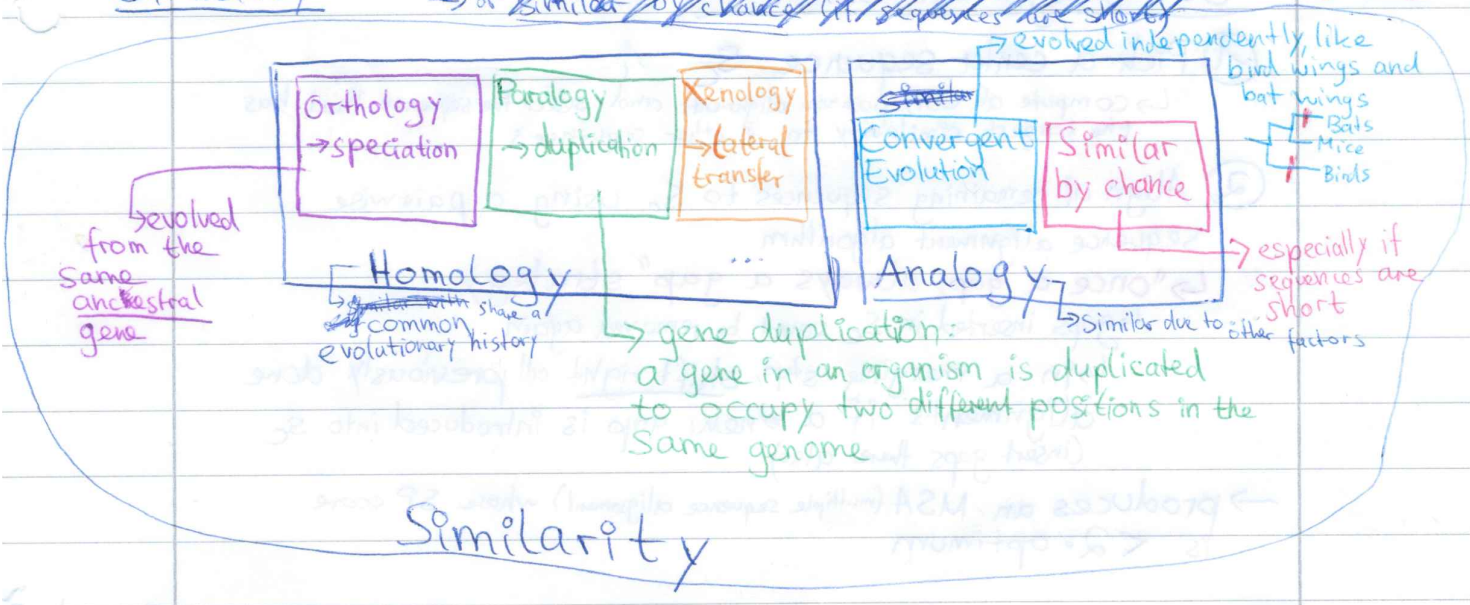
~~Orthology vs Homology~~

~~- orthologous sequences: Sequences in different species that have evolved from the same ancestral gene~~
~~⇒ share a common evolutionary history~~

~~homologous sequences: Sequences that do not share a common evolutionary history, evolved independently~~
~~⇒ convergent evolution like bat wings & bird wings~~
~~⇒ similar by chance (if sequences are short)~~



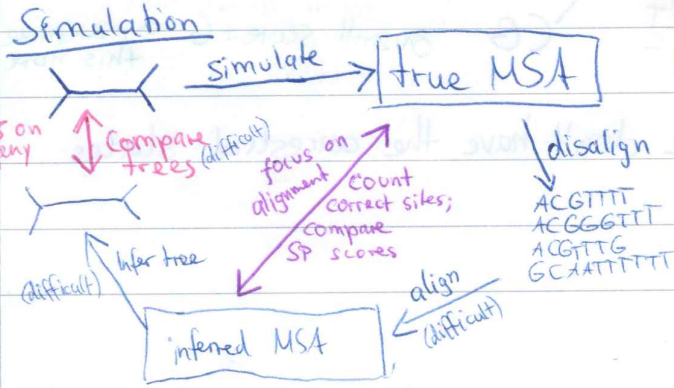
Similarity



Given: n orthologous sequences of different length

Wanted: optimal alignment (allowed to insert gaps, but no columns/sites that consist entirely of gaps, corresponding (homologous) characters are aligned to each other)

- ~~the~~ **Alignment Quality**: ~~unclear definition~~ different criteria
  ↳ SP (sum of pairs) measure, real data benchmarks, evolutionary measures, simulations



Benchmarks:
Mostly structural protein data that has been manually aligned to reflect the protein structure

The SP-measure

- Sum of pairs score: sum of all pairwise scores between characters of a site; favor matches, penalize mismatches and gaps

example with edit distance for scoring:

s1:	A	A	G	A	A	-	A	A
s2:	A	T	-	A	A	T	G	B
s3:	C	T	G	-	G	-	G	C
	2	+ 2	+ 2	+ 2	+ 2	+ 2	+ 3	= 17

$p(-,-) := 0$ ← no penalty for gap↔gap.

↳ same as sum of pair-wise edit distances.

↳ depends on choice of scoring function, not granted that it is biologically plausible

- NP-complete problem to compute SP-score ($O(m^n)$ via DP-Algorithm if all n sequences have length m)

- Star Alignment Heuristics

① Pick a center sequence S_c

↳ compute all $O(n^2)$ pairwise alignments and select the sequence that has the largest similarity to all other sequences

② Align all remaining sequences to S_c using a pairwise sequence alignment algorithm

↳ "once a gap, always a gap" strategy

↳ gaps inserted in S_c cannot be removed again

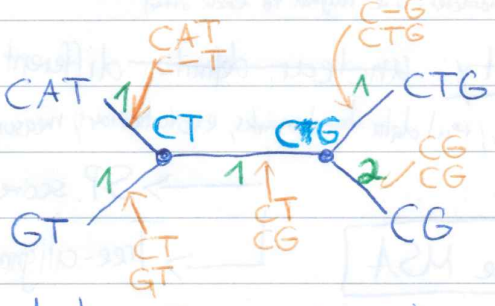
↳ in a merging step, shift right all previously done alignments if a new gap is introduced into S_c (insert gaps there also)

→ produces an MSA (multiple sequence alignment) whose SP score is $< 2 \cdot$ optimum

- Tree Alignment: if an evolutionary tree for the sequences is available

↳ Find an assignment of sequences to the inner nodes such that the sum over the similarity scores on all branches is maximized

example:



$p(a,b) := \begin{cases} 1, & a=b \\ 0, & a \neq b \\ -1, & b = - \text{ (gap)} \end{cases}$

overall score = 6 → maximize this score

- NP-hard because we don't have the ancestral states

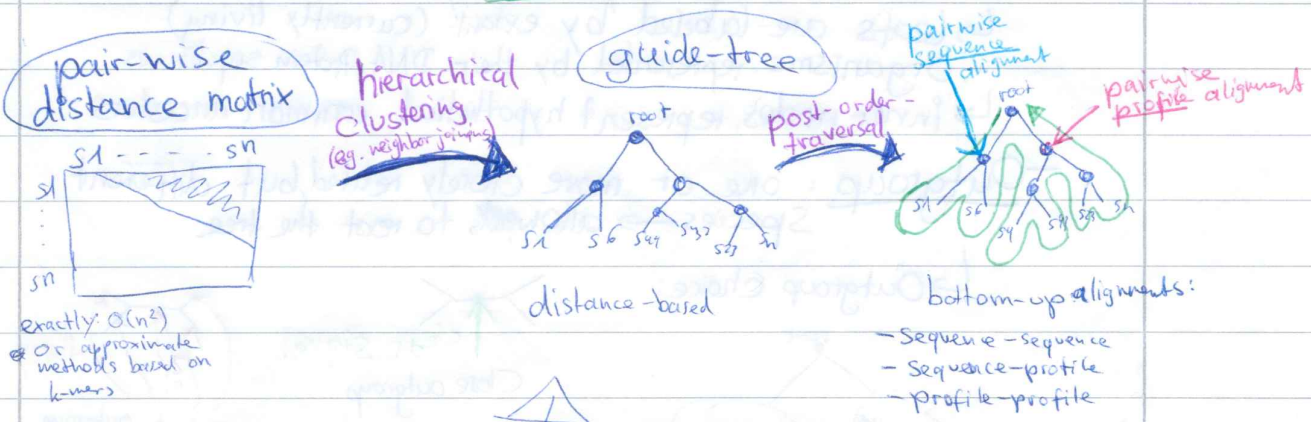
15136

Practical approaches for MSA

~~Tree-Based Alignment~~

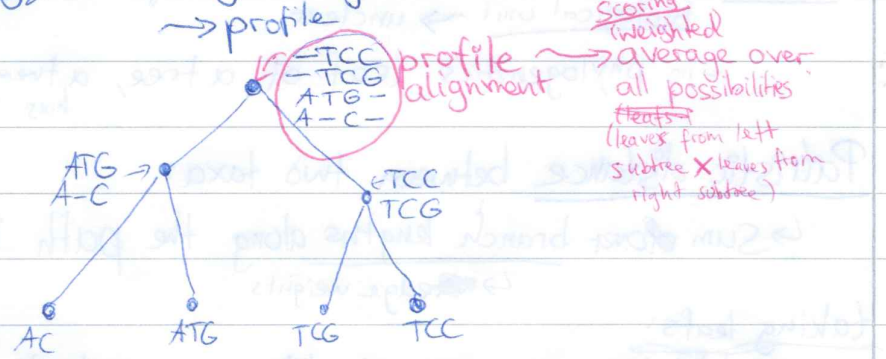
- Tree-Based Alignment is a hen and egg problem
  - ~~tree~~
  - MSA ⇄ tree (needs correct MSA / needs correct tree)
  - ↳ Simultaneous inference of tree and alignment
    - ↳ NP-hard

guide-tree based approach



↳ Progressive MSA

Profile ~~alignment~~: Merge alignments of the two descendant nodes



- problem with progressive MSA: initial pair-wise alignments are "frozen"
  - ↳ every error is propagated through to the final result
  - → can't be corrected when new evidence emerges

↳ Iterative Progressive MSA:

↳ execute progressive MSA several times to refine the alignment,
e.g. MUSCLE Re-Finement: ~~#~~ can return to previous alignments and change them

- Motif-based approaches: Find small motif (substring) common to all sequences
  - ↳ shift sequences such that the motifs are "in alignment"
  - ↳ align regions around these motifs using for instance progressive alignment

16/36

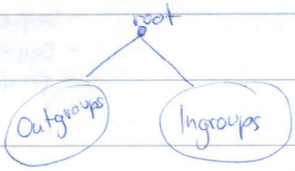
Phylogenetics

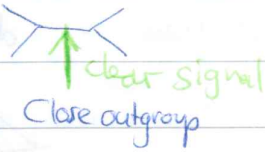
Taxonomy: classification of (living) organisms by looking at shared characteristics (e.g. morphological differences) against the background of biological diversity
↳ characteristics of groups at different hierarchy levels

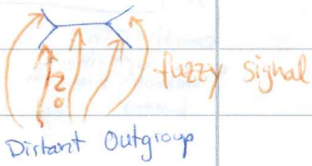
Phylogeny: refers to evolutionary history
↳ unrooted strictly binary tree
↳ leafs are labeled by extant (currently living) organisms represented by their DNA/Protein sequences
↳ inner nodes represent hypothetical common ancestors

- **Outgroup**: one or more closely related, but different species → allows to root the tree

↳ Outgroup choice:





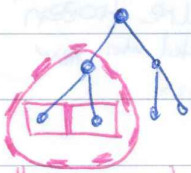


Taxon: -denotes a subtree, one or more species that form a biological unit → unclear
- in phylogenetics: leaf of a tree, a tree with n leaves has n taxa

Patristic distance between two taxa:
↳ sum over branch lengths along the path in the tree
↳ edge weights

taking leafs:

monophyletic

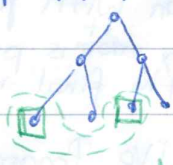


all species share a (direct) common ancestor, all species derived from that common ancestor are included

paraphyletic

all species share a(n) common ancestor, not all species derived from that common ancestor are included

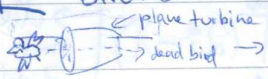
polyphyletic



species do not share an immediate common ancester, excluding other members that would link them

Applications of Phylogenetics

- identifying unknown species
  - ↳ different definitions (~~e.g.~~ for instance by reproduction ability → what about viruses/bacteria?)
  - ↳ phylogenetic placement for identifying anonymous sequences,
  - ↳ phylogenetic species concept: ↳ irreducible cluster, distinct from other such clusters, ~~pair~~ with parental pattern of ancestry and descent
  - ~~Example~~: bird strikes



→ which bird died?

- viral outbreaks: trace back outbreak order
- Snakebites
  - ↳ ~~antivenom~~ same antivenom for closely relates species

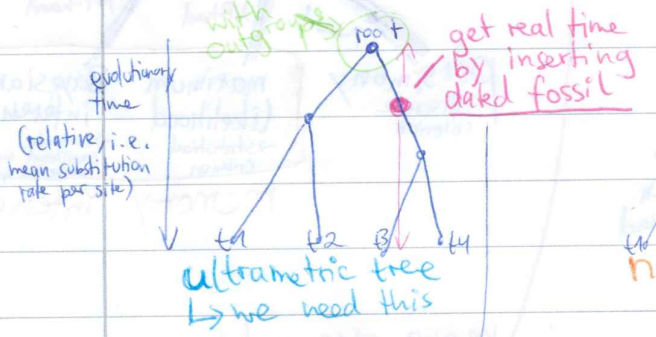
= interpret results from model organisms for other organisms

↳ species that is extensively studied/sequenced to understand particular biological phenomena, with the expectation that discoveries made for this one will provide insight to workings of other organisms

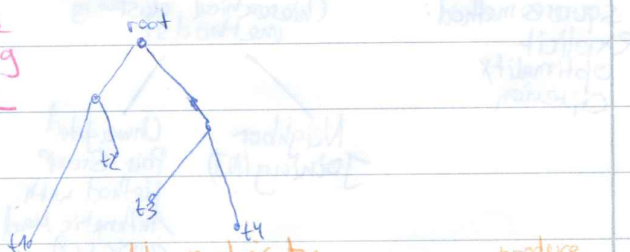
↳ selection criteria: easy to manipulate/grow/extract DNA, economical importance, less ethical problems, ...

↳ examples:
- Escheria coli: gut bacterium → grows fast, ~~easy~~ cheap to cultivate
- Drosophila Melanogaster: fruit fly → breeds quickly
- Arabidopsis Thaliana: flowering plant → small genome
- mouse, rat, chimpanzee, ...

- diversification rates: ~~look at~~ investigate extinction and speciation
- divergence time estimates → for producing ultrametric trees
  - ↳ needed in Tree Dating (dated trees)



ultrametric tree
↳ we need this



non-ultrametric tree
↳ most tree-inference programs produce ~~use~~ this

↳ has no DNA left, either place by empirical knowledge or ~~do~~ use morphological data instead
↳ phylogenetics analysis can has morphological traits as input, too!

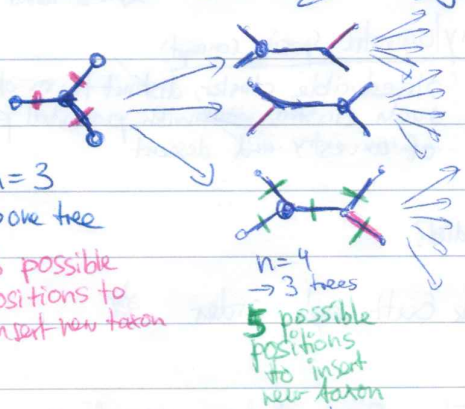
Tree Inference

Tree search problem

Number of unrooted ~~binary~~ trees for n taxa: (unordered)

~~with # node degree #3~~, ~~unordered~~ → each vertex has either 1 or 3 neighbors

~~→ trees that would be binary if we inserted a root~~



n=3
→ one tree
3 possible positions to insert new taxon

n=4
→ 3 trees
5 possible positions to insert new taxon

$$\prod_{i=1}^{n-2} (2i-1)$$

→ die ersten n-2 ungeraden Zahlen

→ this number explodes

- number of rooted binary trees with n taxa
  = number of unrooted "binary" trees with n+1 taxa

- needs MSA, since alignment-free tree inference is typically less accurate → we have not established homology via a MSA

- Building Trees (Phylogeny reconstruction methods)

distance-based methods

→ use MSA to compute matrix of pair-wise distances (may need out-of-core external memory algorithms)
→ build a tree using these distances

- Least squares method: NP-hard
  explicit optimality criterion
- heuristics (hierarchical clustering methods)
  - Neighbor Joining (NJ)
  - Unweighted Pair Group Method with Arithmetic Mean (UPGMA)

less accurate, but faster
→ depend on the accuracy of the distance matrix (must be an exact reflection of the tree)

character-based methods

→ optimality criteria operate directly on MSA & tree
→ take current tree topology & MSA to calculate a score, tells us how well MSA data fits the tree

- Parsimony (NP-hard)
  → discrete criterion
- maximum likelihood (NP-hard)
  → statistical criterion
- Bayesian inference (NP-hard)
  → integrate likelihood over entire tree space

memory intensive

more accurate, but slow

Distance-based methods

- Neighbor Joining : $O(n^3)$

Principle: Given distance matrix of size $n \times n$.

1. Find Minimum
2. Merge Taxa
3. compute new distance matrix of size $(n-1) \times (n-1)$
4. Goto 1 if still more than 1 taxa there.

$O(n^2)$ space, $O(n^3)$ time

3a: Compute average distances $u_i = \sum_j D_{ij} \cdot \frac{1}{n-2}$

3b: Q-Matrix: $D_{ij} - u_i - u_j$ ← find minimum (i,j), there create ancestor x there

$\frac{1}{2} D_{ij} + \frac{1}{2}(u_i - u_j)$ — x — $\frac{1}{2} D_{ij} + \frac{1}{2}(u_j - u_i)$

3c: Update distance matrix:

$$D_{x,k} = (D_{ik} + D_{jk} - D_{ij})/2$$

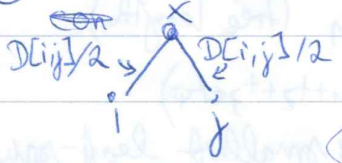
→ does not produce ultrametric trees.

- UPGMA: Unweighted Pair Group Method with Arithmetic Mean : $O(n^2)$

→ can be used if we know that we have an ultrametric tree, usually not the case (for instance higher evolutionary pressure → difficult life conditions)

→ produces rooted, ultrametric trees → misleading results if tree not ultrametric
↳ lineages with shorter lengths are grouped

1. Find Minimum, merge nodes i, j,



2. Update distances: $D[(i,j),k] = \frac{n_i}{n_i + n_j} \cdot D[i,k] + \frac{n_j}{n_i + n_j} \cdot D[j,k]$ $\quad k \notin \{i,j\}$

(n_i: number of elements in node i)

$O(n^2)$ via list of per-column (or per-row) minima

update-list: $O(n)$, look for minimum: $O(n)$

↳ don't need to update the entire matrix each time

Least Squares:

Given: • fixed, fully binary, unrooted tree T with n taxa
• pair-wise distance matrix D

Wanted: assign branch-lengths to the tree such that ~~sum of differences between pair-wi~~ squared

$$Q = \sum_{i<j} (D_{ij} - d_{ij})^2 \stackrel{!}{=} \min$$

- D_{ij}: plain pair-wise distances
- d_{ij}: patristic (tree based) distances
- we can also weight this if we want, e.g. with $w_{ij} := \frac{1}{D_{ij}}$ or $\frac{1}{D_{ij}^2}$

- Q can be minimized by taking derivative and solving system of linear equations in $O(n^3)$
  ↳ with regard to tree-like structure of Q: $O(n^2)$ or even $O(n)$

↳ NP-hard to find the minimum least-squares tree because there are too many possible tree topologies

Minimum Evolution Method: similar to least squares
→ obtain branch lengths via least-squares method
→ minimize total branch length (tree length) of the reconstructed tree ($t_1+t_2+t_3+t_4+\dots$)
↳ do not return the tree with the ~~s~~ smallest least-squares score, but the one with smallest total length out of these.
↳ NP-hard, too.

Character-based methods I

Parsimony: explain the data with the least amount of mutations

Given: • Multiple sequence alignment
• tree topology

Wanted: assignment to inner nodes such that the number of mutations on the tree is minimized

→ similar to tree alignment problem, but here we already have to multiple sequence alignment

→ gaps (also called indels) are treated as undetermined characters (can be A,C,G or T); like "don't-care's" in T1 lecture)

1. Insert virtual root somewhere in the tree → makes computation easier, score stays same
   → we assume that evolution is time-reversible (occured in the same way forward or backward in time)
2. Post-order traversal to compute inner states
3. Compute score on a site-per-site basis
   → assume that sites evolve independently

- Trick: A = 1000, C = 0100, G = 0010, T = 0001

bit-wise AND
↳ if ≠ 0000 → intersection is non-empty
↳ no mutation
↳ else: mutation, count +1

ancestral states are bit-wise OR
↳ union
↳ defer decision which letter to take to a later point

since $n-2$ inner nodes when n taxa; number of sites

$O(n \cdot m)$ space & time-complexity, but with small constant

dynamic programming algorithm

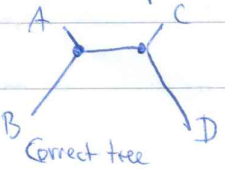
↳ implementation trick: AVX instructions & number of bits set to 1 as built-in operation

→ NP-hard if tree not given

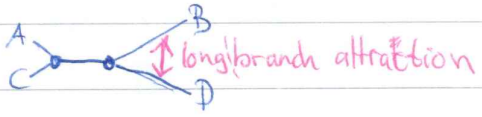
→ returns discrete scores

Long Branch Attraction

↳ because parsimony tries to minimize the number of mutations it faces some problems on trees with long branches



Correct tree



Wrong tree inferred by parsimony

22/136

Tree-Search Algorithms

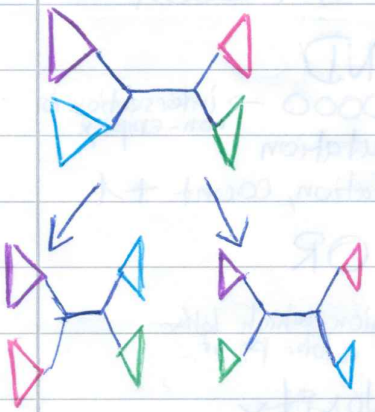
- Obtaining initial starting tree with n taxa
  - ↳ NJ or UPGMA tree → only 1 tree
  - ↳ random tree
    - ↳ start with 3 randomly chosen sequences as leaves
      - ↳ for each other sequence (randomly chosen), chose randomly a branch of the current ~~tree~~ tree to add it there
  - ↳ stepwise addition algorithm (randomized)
    - ↳ for each addition, chose branch that leads to best parsimony insertion score
      - ↳ addition order is randomized

↳ distinct starting trees are useful because of local maxima vs. global maxima

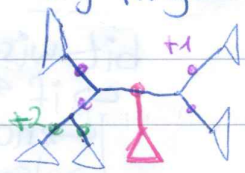
↳ reasonable starting trees: lead faster to a good optimality score than random starting trees
(comprehensive)

Tree Rearrangement
(Basic Moves for changing a tree topology)

Nearest Neighbor Interchange (NNI)

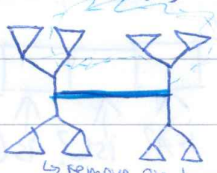


Subtree Pruning and Regrafting (SPR)



↳ remove one subtree
↳ ~~add it back to a branch~~ attach it to another branch

Tree Bisection and Reconnection (TBR)



↳ remove one branch
↳ connect one branch from left one with a branch from right one

- Parsimonator Algorithm
  - ↳ build randomized stepwise addition order parsimony tree
  - ↳ apply SPR-moves to all current subtrees within rearrangement radius of 20
  - ↳ greedy: if an SPR-move leads to a better score, keep it immediately

Maximum Likelihood

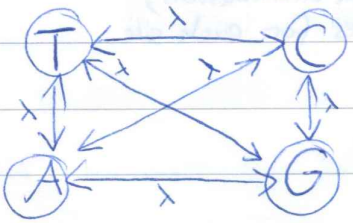
Distances as stochastic processes (Models of Sequence Evolution)

↳ We can have different ways to come from x to y:

$A \to A$
$A \to C \to G \to A \to C \to A$
$A \to T \to G$
⋮

Assumptions:
- Markov process: Probability of $i \to j$ does not depend on what was before i
- Time-continuous: Substitutions can happen at any time
- homogeneous: substitution probabilities do not change in different parts of the tree
- stationary: Overall character frequencies π_i of the nucleotides are at equilibrium and remain constant

Jukes-Cantor (JC)

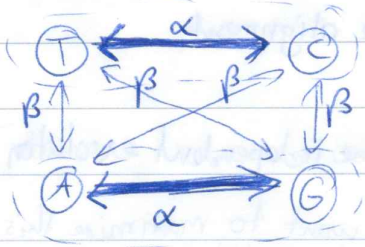


$\lambda_{i,j} = \lambda_{j,i}$

$$Q = \begin{pmatrix} -3\lambda & \lambda & \lambda & \lambda \\ \lambda & -3\lambda & \lambda & \lambda \\ \lambda & \lambda & -3\lambda & \lambda \\ \lambda & \lambda & \lambda & -3\lambda \end{pmatrix}$$

$$\Pi = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$$

Kimura (K)

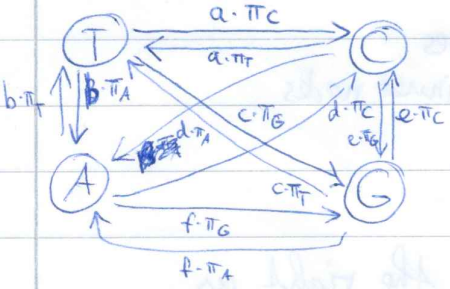


$$Q = \begin{pmatrix} -(\alpha+2\beta) & \alpha & \beta & \beta \\ \alpha & -(\alpha+2\beta) & \beta & \beta \\ \beta & \beta & -(\alpha+2\beta) & \alpha \\ \beta & \beta & \alpha & -(\alpha+2\beta) \end{pmatrix}$$

$$\Pi = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$$

α: transition rate
β: transversion rate

General Time Reversible (GTR)



$\Pi = (\pi_A, \pi_C, \pi_G, \pi_T)$

Mutation in each direction is equally likely: $P(i \xrightarrow{t} j) = P(j \xrightarrow{t} i)$
↳ likelihood computation does not depend on where the tree is rooted

time reversible:

$$\pi_i \cdot \lambda_{ij} = \lambda_{j,i} \cdot \pi_j$$

(λ_{ij}, $\lambda_{j,i}$: entries of Q-matrix)

Q: substitution rate matrix (~~transition rate matrix~~)
↳ each row sums to zero ↳ provides substitution rate between pairs of nucleotides for an instant time dt

Π: stationary distribution (equilibrium frequencies)
↳ sum up to 1

t is a measure of branch length, not time!

transition matrix

$$P(t) = e^{Q \cdot t}$$

$P_{i,j}(t) \triangleq$ probability to end in $\triangleq$ probability of state j, starting from state i, after time t

i → j

if Q is diagonalizable

Trick: $P(t) = e^{Q \cdot t} = e^{U \Lambda \cdot U^{-1} \cdot t} = U \cdot e^{\Lambda \cdot t} \cdot U^{-1} = U \cdot \Lambda^k \cdot U^{-1}$ → easy to compute

↳ spectral decomposition

(Λ: diagonal matrix with eigenvalues of Q; U: eigenvectors of Q as columns)

Rate Heterogenity

↳ different mutation rates at different sites

$$P(t) = e^{Q \cdot u_i \cdot t}$$

↳ draw these rates from a gamma distribution
↳ introduces 1 additional parameter (shape α)

~~↳ average over~~
~~↳ draw 4 rates f~~
↳ Gamma model: draw 4 rates from gamma distribution, ~~for each site~~ average over these 4 rates for each site

Protein Substitution Models

↳ too many free parameters
↳ use precomputed empirical models
↳ count substitutions ~~in matrix~~ in alignments of closely related sequences (no A→T→C happens)

Obtain base frequencies:
↳ count the occurrences of states in the alignment

Likelihood Function ← assume independent evolution

$$LH(T|D) = P(D|T)$$

Tree, Data

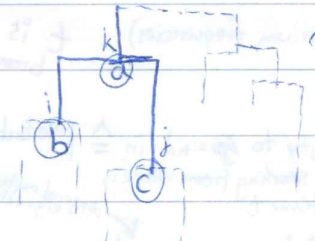
$$= \prod_{S_i \text{ sites}} P(S_i|T)$$

we want to maximize this (find best parameters → Substitution rates, equilibrium distribution, gamma shape, branch lengths)

$$\log LH(T|D) = \sum_{S_i \text{ sites}} \log(P(S_i|T))$$

Given: tree topology, branch lengths (our t's), leaves
↳ sum over all possibilities for inner nodes to get the likelihood

Felsenstein-pruning-algorithm:
↳ move the summation signs to the right as far as possible (place them directly before the related terms)
↳ dynamic programming algorithm, computes the partial likelihoods bottom-up by post-order-traversal of the tree



$$P(T_k \mid \text{node at } k \text{ is } a) = \sum_b P(T_i \mid \text{node } i \text{ is } b) \cdot P(k \to i) \cdot \sum_c P(T_j \mid \text{node } j \text{ is } c) \cdot P(k \to j)$$

Optimizing Parameters

Continuous optimization:
1. Optimize Parameters of the substitution model
2. Optimize branch lengths of the tree → $t_{next} = t - \frac{F'(t)}{F''(t)}$
   - ↳ fix 1, do Newton-Raphson method
   - ↳ typically with iterative method, each branch is optimized seperately → since $P(t) \overset{(1)}{=} U \cdot \Lambda \overset{(2)}{\cdot} e^{\Lambda t} \cdot U^{-1}$ usw.
3. Search over all possible tree topologies → difficult (NP-hard discrete optimization)
   - ↳ do tree rearrangement (NNI, SPR, TBR) → Iterated Local Search

Discrete Operations on Trees

- tree sets
  - ↳ trees with same parsimony score
  - ↳ different trees for different genes / from different databases
  - ↳ ...

Phylogenetic bootstrap

↳ we want support values for each branch
↳ supported by x % of the data

- evolution: stochastic process that yields a multinomial distribution of evolutionary events (→ alignment sites)
- bootstrap new replicate alignments from original alignment
  - ↳ alternative version of the original alignment
- does not introduce new sites that have not been observed

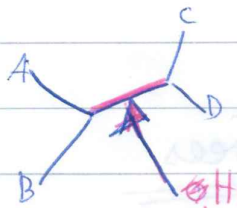
↳ resampling

→ test if the same nodes are recovered → interpretation of bootstrap by simulation:
support value ★ ≥ 70%
↳ ★★ ≥ 95% correct relationship

26/36

ML tree with bootstrap support values

1. Infer ML tree on full alignment
2. Create bootstrap tree set
3. extract bipartitions from ML tree and bootstrap tree set
4. annotate ML tree with relative frequency of its bipartitions in the bootstrap tree set



How many percent of the bootstrap trees do have this bipartition?

Bipartitions

→ inner branches are bipartitions

AB|CDEF → taxa A, B more closely related to each other than taxa C, D, E, F

→ outer branches are trivial bipartitions

A|BCDE → contained in every tree with taxa A, B, C, D, E
→ not informative

- set of bipartitions ⇝ ? → sometimes no tree → tree
  ← unique

- Incompatible bipartitions
  ↳ e.g. ABC|DEF and AD|BCEF

bipartitions $b_1 \equiv B|\overline{B}$ and $b_2 \equiv C|\overline{C}$ are compatible

$\Leftrightarrow (B \cap C = \emptyset) \vee (B \cap \overline{C} = \emptyset) \vee (\overline{B} \cap C = \emptyset) \vee (\overline{B} \cap \overline{C} = \emptyset)$

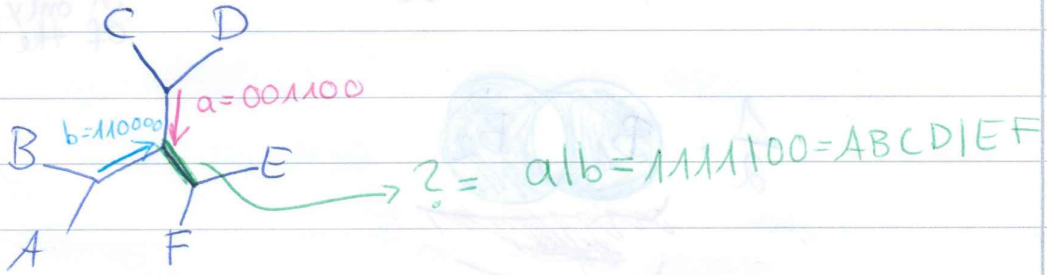
↳ at least one of the four possibilities to pair them is empty.

→ each pair of bipartitions that occur in more than 50% of the trees is compatible

- Store bipartitions
  ↳ number of elements n fixed
  ↳ vector of n boolean values
  ↳ bit-wise operators
  $a \cup b = a \mid b$
  $a \cap b = a \,\&\, b$

Extract bipartitions from tree

↳ via hashing → store bipartitions in hashtable
↳ post-order traversal
↳ compute bipartition from adjacent bipartitions
↳ r[]: one random number per taxon
↳ hash: xor-value of all corresponding random numbers in array



hash(a) = r[C] xor r[D]
hash(b) = r[a] xor r[B]
hash(?) = hash(a) xor hash(b)

Consensus Tree

→ extract bipartitions from tree set, ~~determine~~, count percentage of occurences

majority-rule consensus tree

only bipartitions with freq > 50%

strict majority-rule consensus tree

only bipartitions with freq = 100%

extended majority-rule consensus tree

1. compute majority-rule consensus tree
2. greedily refine tree with non-consensus bipartitions, starting with most frequent one:
   - check if can be added, if so, do so, else proceed to next one

↳ optimal solution is NP-hard

Robinson-Foulds Distance (RF distance) between trees

Given:
- Unrooted tree T_1 with bipartitions B_1
- Unrooted tree T_2 with bipartitions B_2

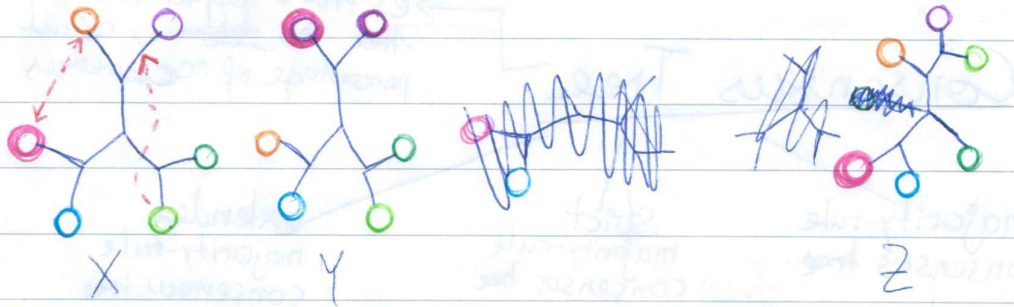
$$RF(T_1, T_2) = \frac{|B_1 \cup B_2| - |B_1 \cap B_2|}{2}$$

$|B_1 \cup B_2| - |B_1 \cap B_2|$ ← number of bipartitions that appear in only one of the trees

$\frac{1}{2} \cdot$ 

$B_1 \Delta B_2$

- RF-distance is a metric
- The RF-distance is not ultrametric:
  ↳ it does not hold that $d(x,z) \leq \max(d(x,y), d(y,z))$



$d(X,Y) = 2$
$d(X,Z) = 2$
$d(Y,Z) = 3$

absolute RF-distance: $\frac{B_1 \Delta B_2}{2}$

relative RF-distance: $\frac{B_1 \Delta B_2}{2 \cdot (n-3)}$

MC MC Methods

- Difference between ML and MCMC

in common:- both re-evaluate the phylogenetic likelihood over and over again for different tree topologies, branch lengths and model parameters
↳ sperd most of their total running time on likelihood calculations on trees

difference:

Bayesian methods (MCMC) → bayesian approach
↓
sample the posterior probability distribution
→ try to maximize posterior probability
↓
rather integrate over the likelihood
→ no numerical optimization needed for model parameters and branch lengths → ~~Hard randomly~~ are proposed randomly
→ easier under complex models
→ has a prior belief

Maximum likelihood (ML) → frequentist approach
↓
find point estimate that maximises the likelihood

better: Jeffrey's prior
↳ non-informative
if we have no idea
for example: uniform priors
→ background knowledge

does the alignment fit the tree and model parameters?

Bayes Theorem

$$P(\text{Tree, Params} \mid \text{Alignment}) = \frac{P(\text{Alignment} \mid \text{Tree, Params}) \cdot P(\text{Tree, Params})}{P(\text{Alignment})}$$

- $P(\text{Tree, Params} \mid \text{Alignment})$: Posterior Probability
- $P(\text{Alignment} \mid \text{Tree, Params})$: likelihood
- $P(\text{Tree, Params})$: prior probability
- $P(\text{Alignment})$: marginal probability → ouch

~~Monte Carlo: pick random ~~50~~ points, average over them, to ap~~

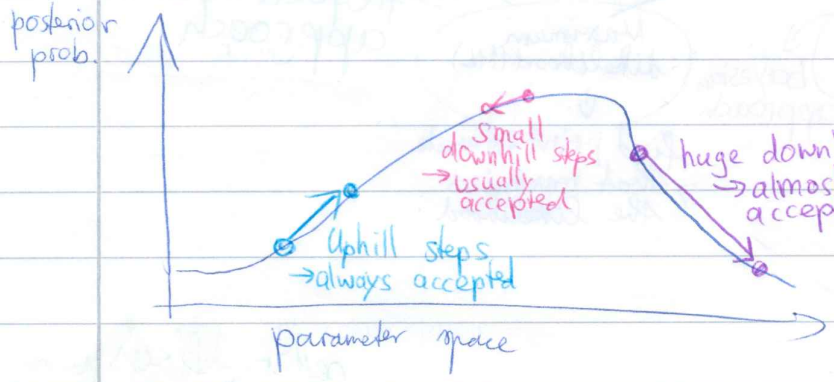
- Monte Carlo Methods: randomly sample data-points in the huge parameter space to approximate the integral

- Markov Chain Monte Carlo

↳ we are interested in areas of high posterior probability → get there faster

↳ biased random walks → spend more time in interesting regions

Robot Metaphor



accept/reject proposals: from point1 to point2

correction factor

acceptance ratio: depends on

$\frac{P(\text{point2}|\text{data})}{P(\text{point1}|\text{data})} \rightarrow \frac{P(\text{point 2})}{P(\text{point 1})} \cdot \frac{P(\text{data}|\text{point 2})}{P(\text{data}|\text{point 1})} \cdot \frac{Q(\text{point 1}|\text{point 2})}{Q(\text{point 2}|\text{point 1})}$

↑ ratio of posterior densities

↳ marginal probability cancels out

prior ratio

likelihood ratio

hastings ratio

↳ needed if proposal distribution is asymmetric

↳ ensures that the probability going from point 2 to point 2 is the same as the probability going from point 2 to point 1, regardless of the proposal distribution

Q ≙ proposal probability

Target distribution vs. proposal distribution → how far left/right will we usually go?

posterior distribution we are trying to sample (integrate over)

↳ decides which point (how far/close) in the landscape to randomly go to/try next

good mixing: find a proposal distribution that requires few steps until ~~conver~~ MCMC chain converges

↳ balance pros and cons

→ very small steps; less refusals

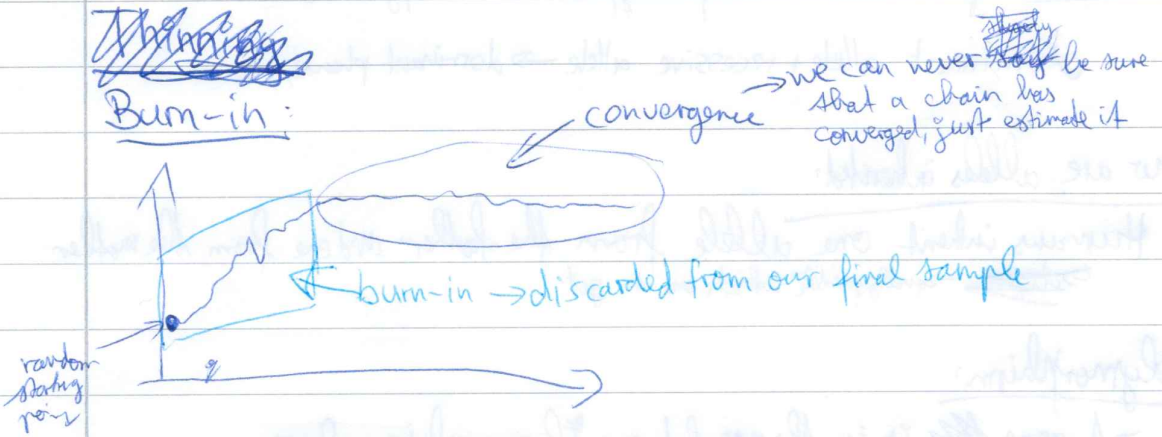
→ larger steps, more refusals

Metropolis-Coupled Markov Chain Monte Carlo

↳ multiple Markov Chains run simultaneously
  ↳ Cold Chain: sees landscape as it is → (1)
  ↳ Hot Chain: sees a melted/flatter → several version of the landscape
    → moves more easily between peaks → larger steps
  - only the cold chain samples

↳ cold robot can swap positions with the hot ones if the hot robots found some better places
→ easier to escape local maxima

Burn-in:



→ we can never be sure that a chain has converged, just estimate it

MCMC in practice

(1) Start with random tree with random branch lengths and random substitution parameters

(2) Iterate using the following steps:
  ~~Propose a change to the tree.~~
  - Propose either
    - a new tree topology
    - a new branch length
    - a new substitution parameter
    and re-calculate the likelihood
  - calculate acceptance ratio of the proposal, accept change or reject it
  - print the current tree with current branch lengths to file only every k (e.g. 1000) iterations → thinning
    ↳ generates a sample from the chain, avoids writing TBs of files

Summarize samples: using means, histograms, credible intervals, Consensus trees, ...

2136

Population Genetics

Biological Evolution: Change of the frequency of occurrence of features of individuals of the population should be inherited

4 Evolutionary forces

↳ Genetic Drift: Chance (other than a random mutation)

↳ Migration

↳ Mutation

↳ Natural Selection: Response to some pressure (e.g. antibiotics, climate change)

Genotype: Full hereditary information

Phenotype: actual observed propertys → expression

Dominant: produces a dominant phenotype → one copy is enough

Recessive: produces a recessive phenotype → both copys have to be it

dominant allele + recessive allele → dominant phenotype

How are alleles inherited:

Humans inherit one allele from the father and one from the mother

~~diploid~~ - equally probable which one to get

Polymorphism:

- A gene is in the population polymorphic when there exist multiple alleles

SNP: Single Nucleotide Polymorphisms ("Snips")

↳ polymorphic sits in the MSA of individuals (have more than one nucleotide state)

Hardy-Weinberg Model

Assumptions

- infinite population size
- random mating, no natural selection, no migration
- diploid population
- gene A with 2 alleles: A and a
- Current frequencies (at generation 0) of allele pairs defining the genotype

$f(AA) = x$
$f(Aa) = 2y$
$f(aa) = z$

- Current frequencies of gametes (single alleles) at generation 0:

$f_0(A) = x + y$
$f_0(a) = z + y$

- Sexes have same distribution of the 3 genotypes AA, Aa, aa

Punnett square

	A	A
A	AA	AA
a	aA	aA

offspring genotype possibilities

	Females A(p)	Females a(q)
Males A(p)	AA (p^2)	Aa ($p \cdot q$)
Males a(q)	Aa ($q \cdot p$)	aa (q^2)

$p^2 + 2pq + q^2 = 1$

↳ if we filter out all evolutionary forces and assume infinite population size, the frequencies of the alleles stay the same.
↳ Hardy-Weinberg Equilibrium

↳ If we see change in allele frequencies between parents and offspring → evolution happened

Wright-Fisher model

↳ Populations are of finite size 2N
↳ non-overlapping generations (e.g. annual plants)
↳ constant population per generation
↳ each individual from offspring generation picks a parent at random from previous generation, as uniformly distributed, each parent can be picked more than once
↳ each offspring inherits information from the parent

↳ binomial random sampling
↳ probability to pick allele A as parent is $\frac{\#A}{2N}$.
↳ binomial distribution of alleles

$$\binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}$$

absorbing state:
↳ probability to escape this state is zero
→ if allele frequency is 0 or 1.
↳ **fixation**: probability to end up in $f(A)=1$
↳ $= \frac{\#A}{2N}$

Random genetic drift:
Change in allele frequencies over generations in finite populations due to re-sampling

Heterozygosity:
~~Probability that two alleles are different~~

$$Het_t = Het_{(t-1)} \cdot \left(1 - \frac{1}{2N}\right) \quad \hookrightarrow Het_t = Het_0 \cdot \left(1 - \frac{1}{2N}\right)^t$$

($\frac{1}{2N}$: Probability that two randomly chosen alleles are different)

↳ expected probability that an individual will be heterozygous → proportion of individuals heterozygous at a locus from original proportion
↳ measure for gene diversability

↳ $-\frac{1}{2N} \cdot Het \doteq$ loss of heterozygosity per generation due to genetic drift

Positive selection:

if natural selection has favored #allele A
↳ reproduces more often

Mutation-drift balance

↳ genetic drift removes polymorphisms
↳ mutation introduces polymorphisms

rate of evolution: the probability of a new mutation to arise in the population and to eventually become fixed

μ: probability of mutation per generation and individual

$2N$ individuals → $2N\mu$ mutations per generation

- probability that a mutation will be fixed is $\frac{1}{2N}$

↳ rate at which a mutation will arise and fix is

$$\frac{1}{2N} \cdot 2N\mu = \mu$$

→ We want $\Delta Het \doteq 0$

gain in heterozygosity due to mutation:

↳ $2\mu \cdot (1 - Het)$

2μ: mutation rate; at least one of the two mutates

$(1 - Het)$: homozygosity

$$Het_{t+1} = Het_t - \frac{1}{2N} \cdot Het_t + 2\mu \cdot (1 - Het_t)$$

Het_t: initial; $\frac{1}{2N} \cdot Het_t$: due to genetic drift; $2\mu \cdot (1 - Het_t)$: due to mutation

Phylogenetics vs. Population genetics

↳ Study of Polymorphisms in a population → why? how do? Can it get away?

↳ given data (MSA of individuals of the same species)
↳ simulate evolution under different scenarios
↳ compare if one of the scenarios fits the summary statistics (e.g. # SNPs) of our empirical dataset
→ help us to see if positive selection occured, help, allow parameter estimation of population
↳ Evolution at different scales
↳ less mutations ⇒ correctness of MSA more important

↓ inheritance of a trait

↳ how the frequency of alleles which control the trait change over time

Introduction to Bioinformatics
Biological Background
Genome
Chromosome
Gene
Exons
homozygous & heterozygous
haploid
diploid
DNA
coding/non-coding
5' to 3' end
transcriptome
RNA
secondary structure
types of RNA
mRNA
rRNA
tRNA
16S gene
Eukaryota vs. Prokaryota
3 domains of life
protein synthesis
alternative splicing
DNA/AA alphabet
start/stop codon
model organism
codon position
ambiguities
synonymous & non-synonymous substitution
sequencing
paired-end vs. single-end reads
Sanger
next generation sequencing
Pairwise Sequence Alignment
Edit distance
Levenshtein Hamming
BLOSUM substitution matrices
maximize similarity
Global Alignment: Needleman-Wunsch
Local Alignment: Smith-Waterman
Search similar sequences
BLAST
seeding
evaluation
extension
Genome Assembly
de novo
de Bruijn graphs
contigs → scaffolds
by reference
hashing-based
BWT-based
better
Multiple Sequence Alignment
simulation
Homology vs. Analogy
Orthology → speciation
Paralogy → gene duplication
convergent evolution
similar by chance
quality
SP-score
NP-hard
star alignment heuristic
Tree-based
practical approach
infer both simultaneously
progressive MSA
guide-tree via hierarchical clustering
post-order traversal
iterative progressive MSA
motif based
Population Genetics
evolutionary forces
genetic drift
mutation
migration
natural selection
positive selection
Polymorphism, SNPs
Mendelian genetics
mutation-drift balance
heterozygosity
$Het_t = Het_{t-1} \cdot (1 - \frac{1}{2N})$
Hardy-Weinberg
infinite population size
Wright-Fisher
finite population size
lots of simulations
Phylogenetics
Tree dating
ultrametric
dated fossil
Taxonomy
Phylogeny
outgroups
monophyletic
paraphyletic
polyphyletic
application
identify species
viral outbreaks
diversification rates
divergence time estimates
Tree Inference (needs MSA)
distance based
NJ
UPGMA
heuristics
least squares, minimum evolution criterion
use matrix of pairwise distances
character-based
parsimony
least amount of mutations
long branch attraction problem
ML
maximize P(Data|Tree)
MCMC
maximize P(Tree|Data)
target vs. proposal
prior
robot metaphor
nucleotide substitution models
JC
Kimura
GTR
time reversible
$\pi_i \cdot \lambda_{ij} = \lambda_{ji} \cdot \pi_j$
likelihood doesn't depend on root choice
Discrete Operations on Trees
RF-distance
not ultrametric
Consensus tree
strict
majority rule
extended majority rule
compatible
bipartitions
randomized stepwise addition order algorithm
Tree Rearrangement
NNI
SPR
TBR
Phylogenetic bootstrap