COM SCI C121 Week 4

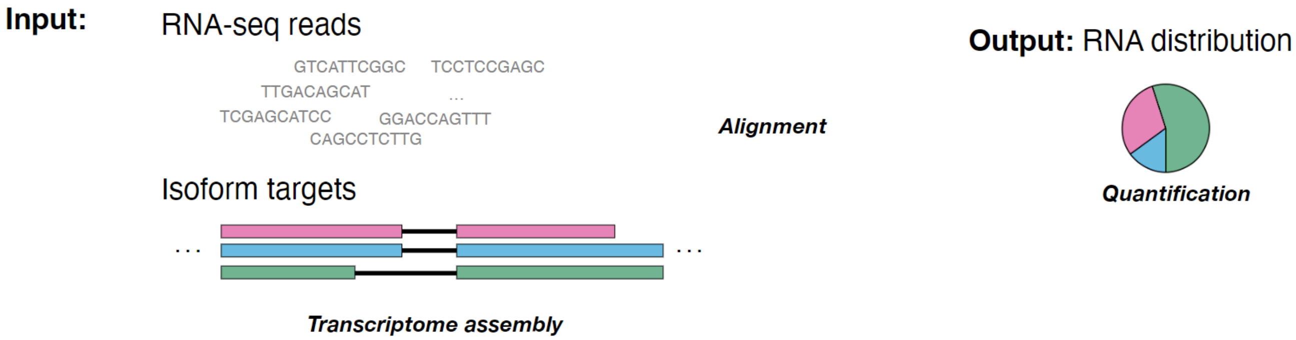
Aidan Jan

April 30, 2024

Pseudoalignment

RNA-seq quantification is a computational problem.

- **Goal:** given a known set of isoform targets (genes) and RNA-seq fragments, recover the distribution of RNA molecules.



Alignment is the (in)exact matching of a subsequence to a reference.

- In simplest terms:
  - chromosome 1, position 342,215
  - transcript A, position 32; transcript B, position 3, . . .
- Also possible:

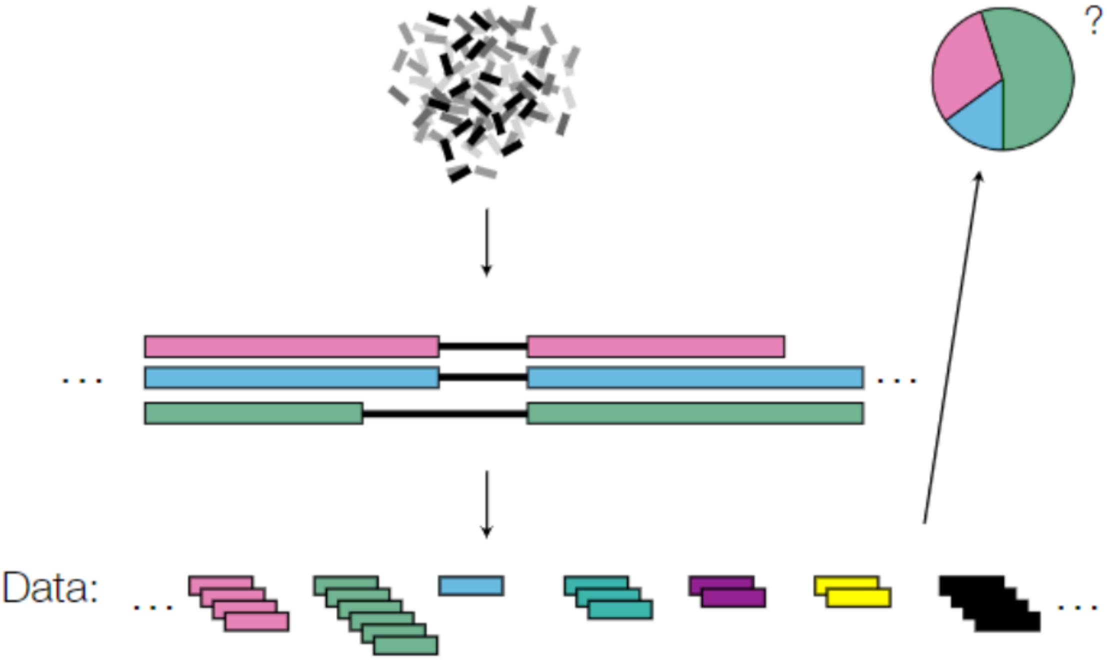
```
TACGGGCCCGCTA-C
TA---G-CC-CTATC
```

A Fundamental Problem: Alignment and Counting

- Classical approaches for exact matching are too slow. (Dynamic Programming, etc.)
  - Boyer-Moore $O(|R| + |T|)$
- Contemporary methods use heuristics
  - Seed and extend
    - To 'seed' means you make an assumption that one of the reads is extremely accurate, and then you assume that read is correct and extend based on the other fragments.
- Our approach: use the *redundancy* and structure of the target sequences

A Fundamental Problem: Counting and Quantification

- **Quantification:** given many alignments to a reference transcriptome, what is the likely *relative* abundance of each isoform?
  - Complication: most reads will give many, many transcripts



Equivalence classes are sufficient for quantification.

- Equivalence classes is the assumption that which isoform the data matches to does not matter.
- Essentially, an equivalence class is the set of isoforms a transcript is compatible with.

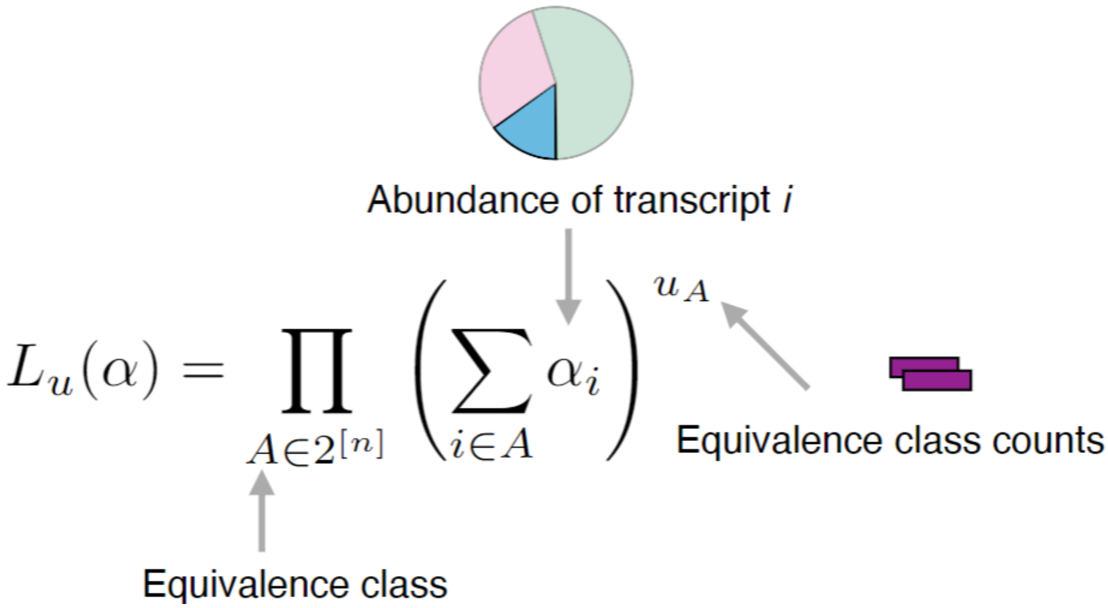
Quick Aside: k-mer

A **k-mer** is a sequence of length k that is a substring of a longer sequence

Consider 'ACGGT':

- k-mers of length 3: ACG, CGG, GGT
- k-mers of length 4: ACGG, CGGT

The Linear Allocation Problem Likelihood

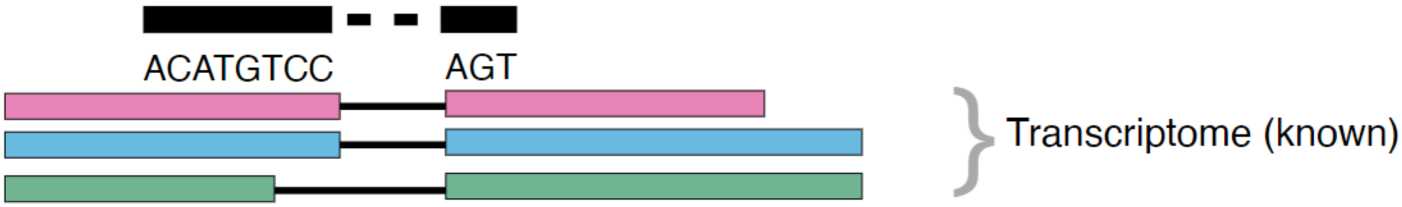


$$L_u(\alpha) = \prod_{A \in 2^{[n]}} \left(\sum_{i \in A} \alpha_i \right)^{u_A}$$

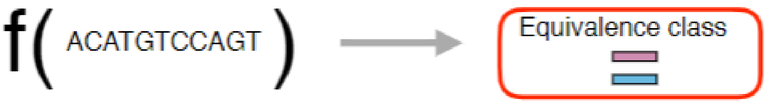
All observed equivalence classes easily fit into the memory on a laptop.

- 100M kmers
- <1M equivalence classes

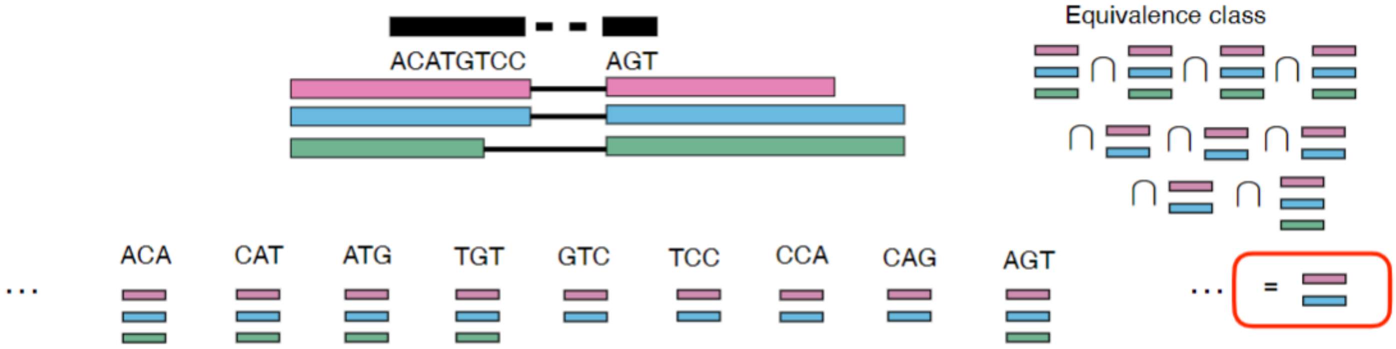
Kallisto: Introducing Pseudoalignment



- **Goal:** provide the "set of colors" (set of transcripts) that an observation could have come from.
- **Pseudoalignment:** a map from the sequence to equivalence class.



A Naive Implementation of Pseudoalignment



- Break the data into k-mers and match them with the transcriptome.
- Optimal length of k-mers is based on many factors; the only way to get them is by experiment. This is not practical to find.
- Note that you have to compare every k-mer here to get the equivalence class. If the read has a thousand base pairs, it is *extremely* slow.

De-Brujin Graphs for Genome Assembly

The goal is to reconstruct target from k-mer perfect samples.

Target: `AAABBBA`

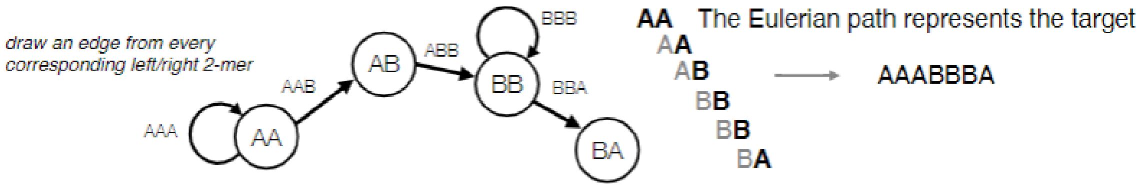
(Sample 3-mers)
↓

AAA AAB ABB BBB BBA

(Form left/right 2-mers)
↓

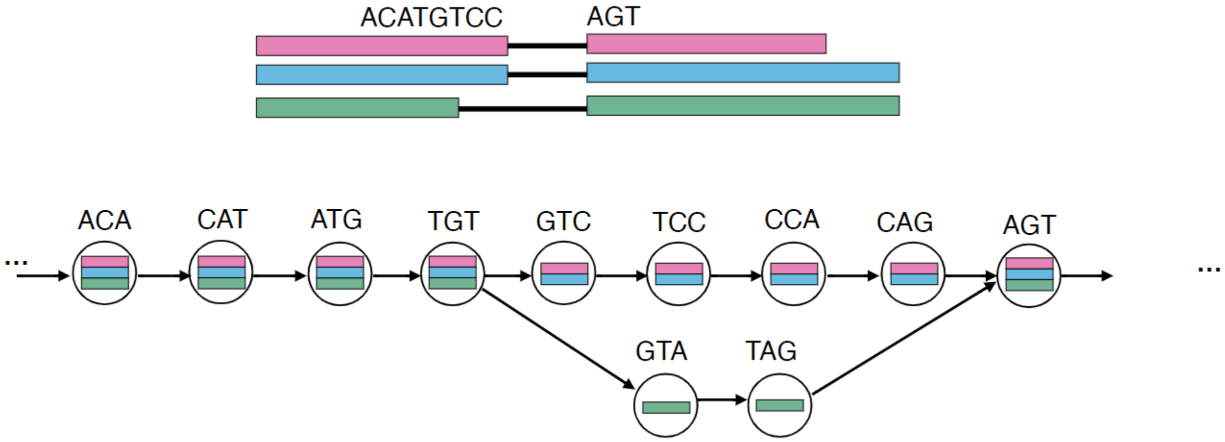
AA / AA AA / AB AA / BB BB / BB BB / BA

(Draw De-Brujin Graph)
↓


draw an edge from every corresponding left/right 2-mer
AAA
AA
AAB
AB
ABB
BBB
BB
BBA
BA
AA The Eulerian path represents the target
AA
AB
BB
BB
BA
AAABBBA

- Normally in Computer Science, we want to use the De-Brujin graph to find the target. However, in this case, going backwards is very, very hard.
- Instead, we assume we know the target and we generate the graph from the target.
- Then, we can use a probabilistic model to check the validity of the data.

Transcriptome as a De-Brujin Graph



- To find the isoforms, we first go to the first node of the read (ACA). This can be done quickly with a hash table containing pointers to the nodes; simply hash the three letters to get the node.
- Next, we jump to the branch point. (In this case, it's TGT branching to GTC or GTA). Based on the read, we have GTC, so it rules out the green isoform.
- We go to the next branch point. (In this case it's AGT branching in from CAG and TAG). AGT contains all three isoforms so it does not give any new information.
- To determine the equivalence class for the read, we take the intersection of the three reads (beginning, first branch, second branch), and we find the equivalence class is the blue and red isoforms.
  - This is considerably more efficient than taking the intersection of every node on the read like before. If there are thousands of nodes but only a couple branch points, we don't have to consider any of the nodes other than the branch points, accelerating the process by up to a few thousand times while still being accurate.
  - This algorithm is called **Kallisto**.