Cufflinks

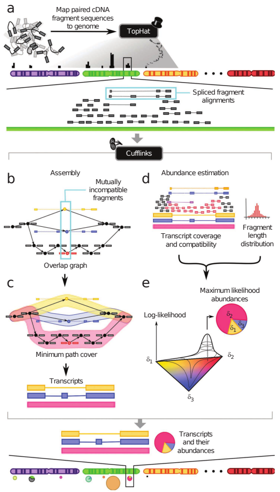
A conceptual, relatively mathematics-free explanation

Ryan Thompson

2011

Outline

Overview


a
Map paired cDNA fragment sequences to genome
TopHat
Spliced fragment alignments
Cufflinks
b
Assembly
Mutually incompatible fragments
Overlap graph
c
Minimum path cover
Transcripts
d
Abundance estimation
Transcript coverage and compatibility
Fragment length distribution
e
Maximum likelihood abundances
Log-likelihood
Transcripts and their abundances

- Reference-based transcript assembler and quantifier
    - Assembles reads by their genomic positions
    - Not directly by sequence
- Assembly and quantification happen separately
    - One could use Cufflinks for one and another tool for the other
- We will explore Cufflinks' approach to both problems

Important Terms

- **Transcript:** An RNA molecule transcribed from DNA
- **Primary transcript:** Transcript that has yet to undergo modification
- **Genomic location:** Pair of coordinates; $5'$-TSS & $3'$ poly-A sites (regardless of internal splicing)
- **Transcriptome:** Set of all transcripts with corresponding abundance values.
- **Locus:** Set of overlapping transcripts that does not overlap any other locus.
- **Abundance:** The fraction of transcript molecules that come from a particular source

Outline

Overview of Assembly

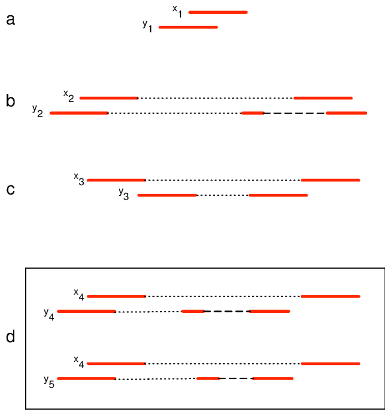
- Each locus is assembled independently of the others
- Algorithm designed to produce an assembly with desirable properties:
  - Every fragment is consistent with at least one assembled transcript
  - Every transcript is tiled (covered entirely) by reads
  - The number of transcripts is the minimum required
  - Abundance estimates are fully determined
- In other words, the minimal and most parsimonious set of transcripts that fully explains the observed reads

Partial Orders and DAGs

- An order is a relation on a set where any two items either have a relationship of "greater than", "less than", or "equal to".
- We can represent a full order by an unbranching number line, since every item is comparable to every other
- A partial order allows any of the above, plus "not comparable to"
- Now, instead of a single line, we have a branching graph – a "directed acyclic graph"
- The partial order is equivalent to the DAG

Partially Ordering Fragments

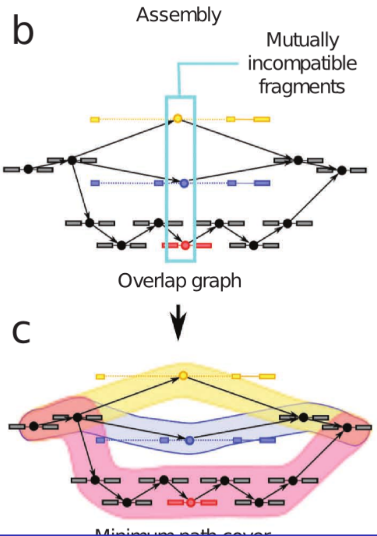
- Fragments are *compatible* if they imply the same splice sites or none (**a**), *incompatible* if they imply different ones (**b**)
    - compatible fragments are ordered by their start positions
- Paired-end fragments can be *nested* (**c**), in which case they are merged into their containing fragments
- Some read pairs are *uncertain*, because they are consistent with multiple splicing patterns (**d**).
    - These are excluded from assembly (but not quantification)



DAG to Assembly

Deriving transcripts from partially-ordered fragments

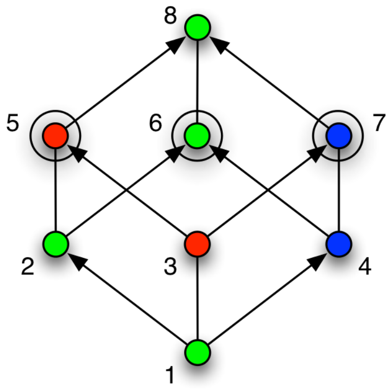
- A *chain* is a “path” through the DAG from source (TSS) to sink (poly-A site)
- Each chain (ideally) represents a transcript isoform
- A partition of the DAG into chains yields an assembly of the locus into transcripts
- We want the minimum number of chains required



Minimum Partition, Maximum Antichain

One algorithm's min is another algorithm's max

- An *antichain* is a set of mutually incompatible elements
- None is upstream or downstream of any other
    - "Perpendicular" to every chain
- The maximum antichain has the same size as the minimum number of chains
    - Think: "maximum branchiness"
- An efficient algorithm exists for finding the maximum antichain, and hence, the required number of transcripts

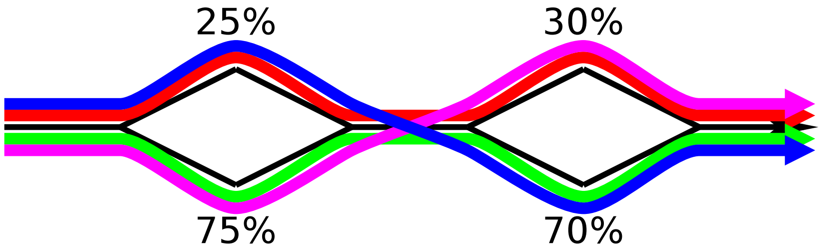


Phasing Distant Exons
Psi vs Psi

- We may have to choose from multiple ways of partitioning the DAG
- Define percent-spliced-in (*psi*, ψ) for a given fragment as the fraction of overlapping fragments that are compatible with that fragment
- ψ is an estimate of the fraction of transcripts that include this fragment
- Define a cost function for assembling two fragments together:
    - $C(y,z) = -\log(1 - |\psi_y - \psi_z|)$
- Key idea: the cost function increases as the ψ values diverge
- Choose the minimum partition with minimum cost

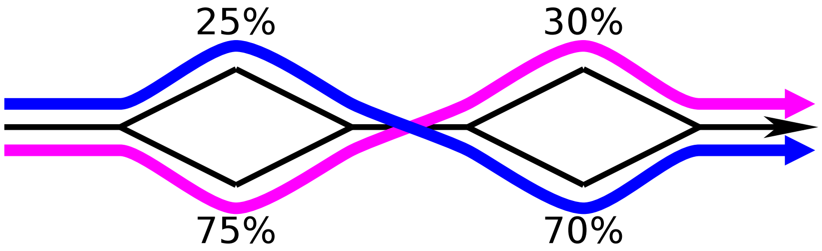
Phasing Example
Stop me if you've heard this one

- Max antichain = 2; must choose 2 transcripts
- Which ones? Either red/green or blue/magenta



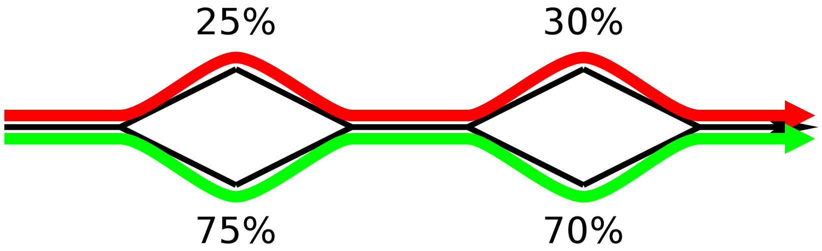
Phasing Example
Stop me if you've heard this one

- blue/magenta phases dissimilar ψ values (bad)



Phasing Example
Stop me if you've heard this one

- red/green phases similar ψ values (good)



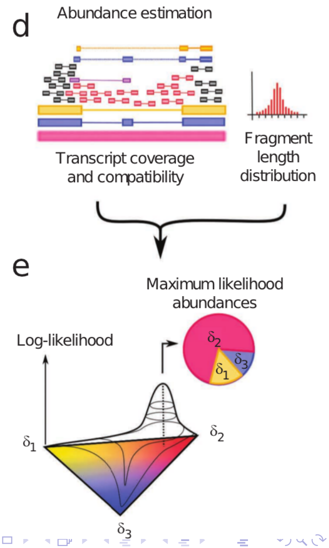
Why Minimize Chain Count?

- In the previous example, there could be 3 or 4 isoforms instead of 2
- But it becomes impossible to quantify all 4 isoforms
    - System with 4 unknowns but only 2 equations
- If we don't choose a minimum partition, the quantification is not uniquely determined
- The chosen chains will likely correspond to the most abundant isoforms

Outline

Defining the Quantification Problem

- Recall a transcriptome is a set of transcripts *with abundances*
- Given a set of transcripts and a set of reads, we want to estimate the abundances
- First we work forward to figure out what set of reads we expect from a given set of abundances (building the model)
- Then we work backward to figure out the abundances that best account for the reads we observe (maximum likelihood estimation, MLE)


d
Abundance estimation
Transcript coverage and compatibility
Fragment length distribution
e
Maximum likelihood abundances
Log-likelihood
δ_1
δ_2
δ_3

Statistical Model
How we think RNA-Seq works

- Assume that fragments are sampled uniformly from all transcripts in proportion to their abundances
  - Could use other distributions, accounting for amplification & sequencing biases
- Define *fragment abundance*: the probability that a fragment selected at random originiates from that transcript
  - Longer transcripts yield more fragments
  - also depends on fragment length distribution
- Fragment abundance is easier to estimate, and can easily be converted to abundance

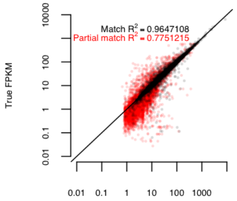
Further Simplification

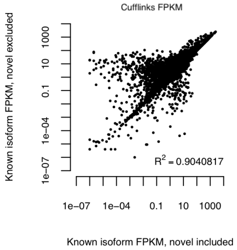
Have mercy on the computer

- Finding fragment abundance is theoretically easy, but numerically hard
- Fragment abundance (α) can be decomposed into locus abundance (β) and conditional abundance of the fragment given the locus (γ): $\alpha = \beta \cdot \gamma$
- Cufflinks estimates β for each locus and γ for each transcript in each locus from the data using MLE
  - $\hat{\beta}$ is just the fraction of fragments mapping to a locus
  - all $\hat{\gamma}$ in a locus are found by expectation maximization
- Then Cufflinks works backward to α and converts to FPKM
- Confidence intervals are estimated from variances of $\hat{\beta}$ and $\hat{\gamma}$

Always Allow Novel Transcripts

- Cufflinks has an option to quantify a set of existing transcripts without assembling any new ones
- Do not use this option
- This will cause inaccurate quantification of known transcripts, since reads that should be attributed to the novel splice forms will instead be assigned to the known ones





Outline

Summary

- Cufflinks is both a transcript assembler and quantifier
- It assembles the most parsimonious minimal set of transcripts that explains all the data
- It quantifies transcripts by solving for the abundance levels most likely to generate the observed data
- It should be quite reliable for at least getting the dominant isoform and estimating gene-level abundance

Practical Optimization

Better cluster-based parallelization of assembly

- Assembly is done independently for each locus
- One could split a dataset into one SAM file per locus and then submit a separate job for each
  - We currently only submit one job per *chromosome*
  - Larger or more complex chromosomes take much longer
- Cufflinks can do this itself using *threads* on one machine, but not on a cluster
- After assembly, concatenate result files and run a single instance of Cufflinks to quantify
- We would need a script to split a sorted SAM file of *paired-end* reads into loci