BC205: Algorithms for Bioinformatics. VII. Genome Scale Algorithms

Christoforos Nikolaou

May 24th, 2018

In previous chapters

- We saw how:
  - We map NGS reads on a reference genome
- And discussed NGS-related problems such as:
  - Reconstructing a genome from scratch by assembling reads (DNA sequencing)
  - Differential Gene Expression Analysis (in RNASeq)
  - Detection of transcription factor/DNA binding proteins peaks (in ChIPSeq)
  - Definition of regions of specific chromatin/epigenetic characteristics (in ChIPSeq)
  - Definition of structurally-related domains in the three-dimensional conformation of chromatin (in HiC-Seq)
- Main Problems we need to solve
  - Scale
  - Complexity (combinatorics)

1. Genome Assembly

- The problem: Given a set of reads, we need to join reads such as the genome of the sequenced organism(s) is reconstructed as accurately as possible. There is NO reference genome
- Solutions: Most solutions are based on the implementation of De Bruijn Graphs, which are based on the original Eulerian path reconstruction problem of the Bridges of Konigsberg.

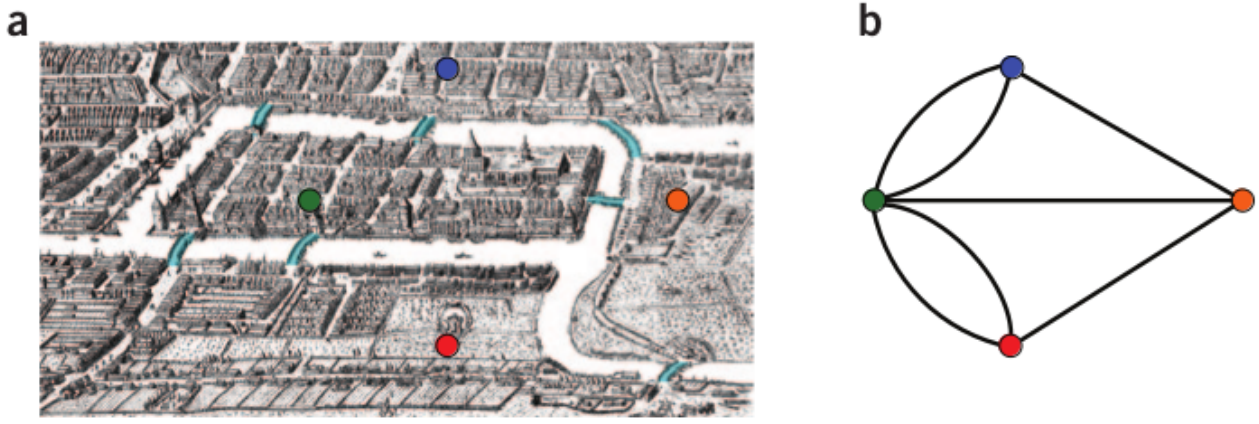


Figure 1 Bridges of Königsberg problem. (**a**) A map of old Königsberg, in which each area of the city is labeled with a different color point. (**b**) The Königsberg Bridge graph, formed by representing each of four land areas as a node and each of the city's seven bridges as an edge.

De Bruijn Graph Reconstruction

- Formulation: Starting from a fully-connected graph, find a path that passes from each node once. In sequence terms: find the shortest 'superstring' that contains all possible 'substrings' of length k (k-mers) over a given alphabet.

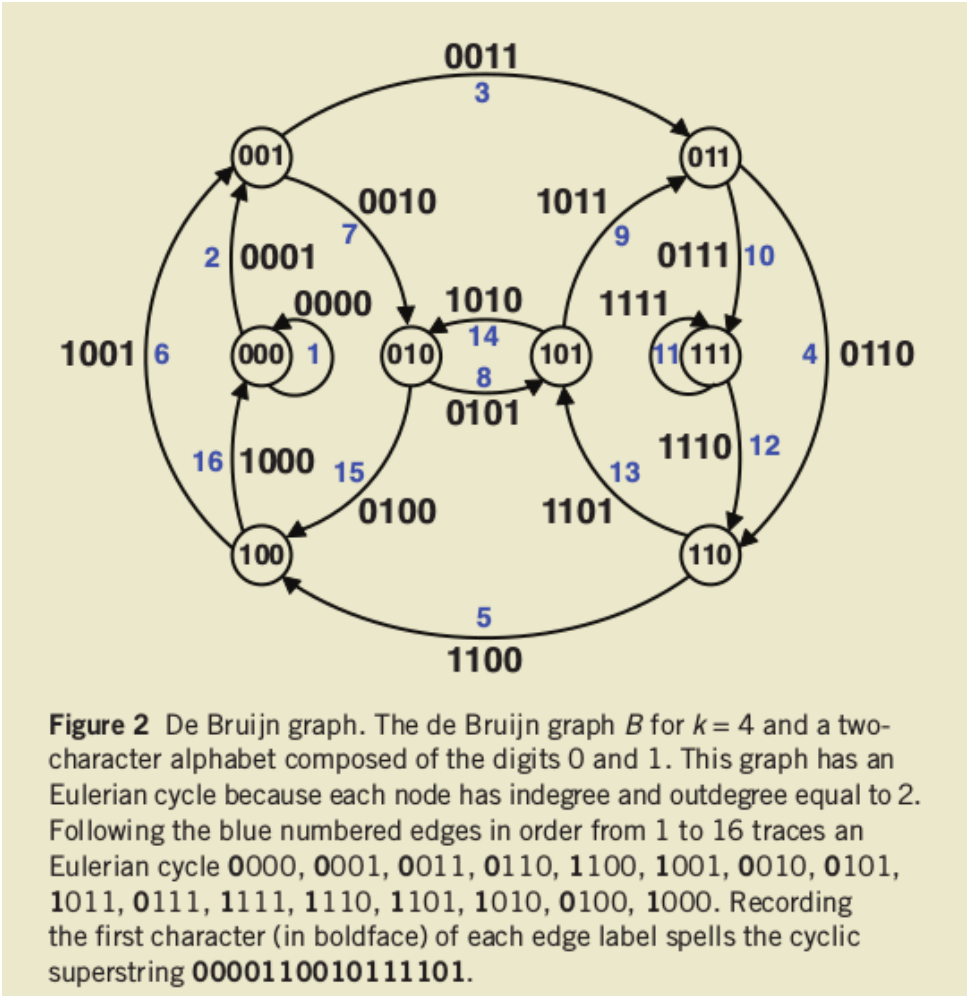


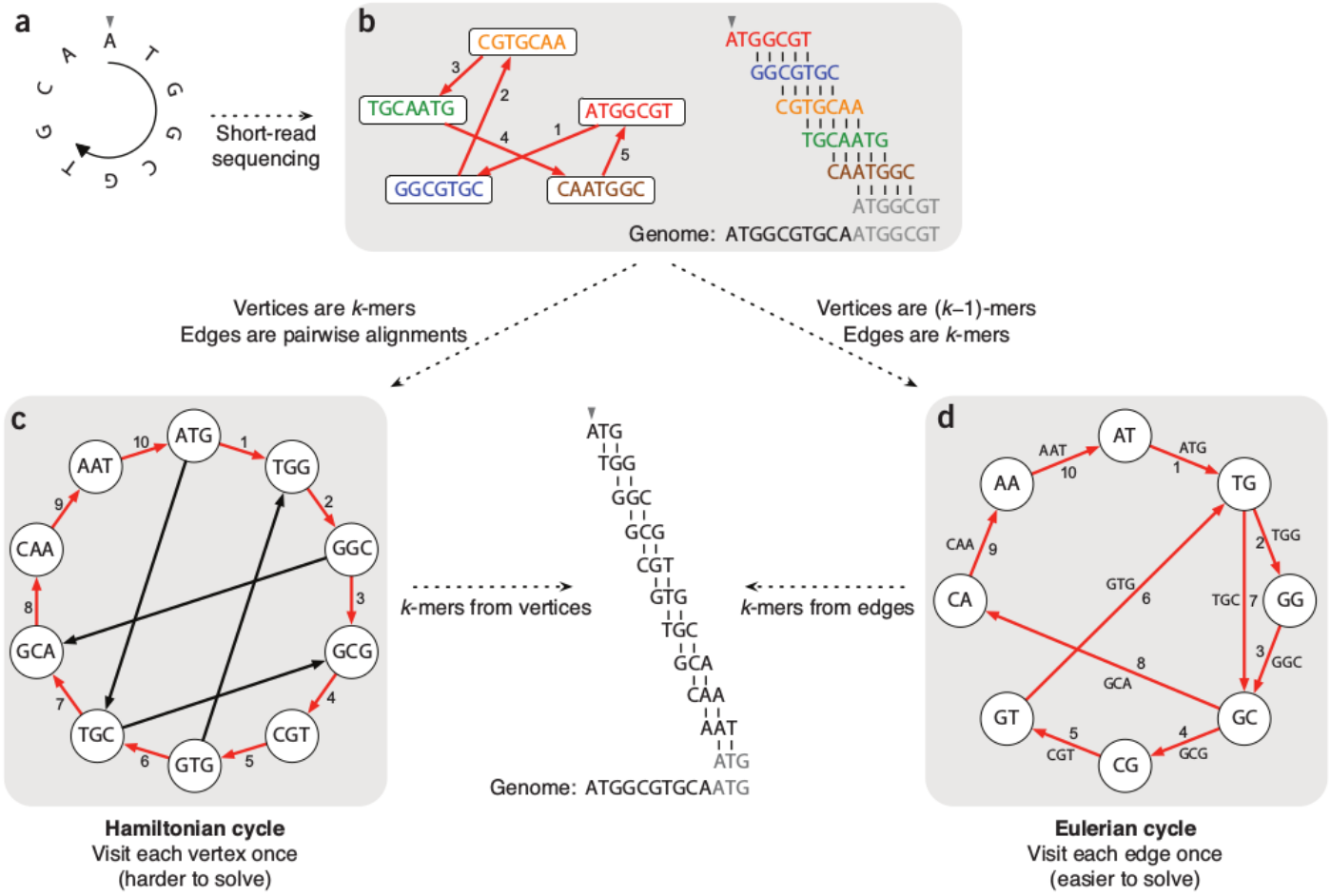
Figure 2 De Bruijn graph. The de Bruijn graph B for $k = 4$ and a two-character alphabet composed of the digits 0 and 1. This graph has an Eulerian cycle because each node has indegree and outdegree equal to 2. Following the blue numbered edges in order from 1 to 16 traces an Eulerian cycle **0**000, **0**001, **0**011, **0**110, **1**100, **1**001, **0**010, **0**101, **1**011, **0**111, **1**111, **1**110, **1**101, **1**010, **0**100, **1**000. Recording the first character (in boldface) of each edge label spells the cyclic superstring **0000110010111101**.

De Bruijn Graph Solution

- The problem: Solving the problem via nodes (finding a Hamiltonian path) is rather complex
- The solution: De Bruijn reformulated the problem in finding an Eulerian path, that is *visiting each* **edge** *of the graph once*. As long as the graph is balanced (in-degree==out-degree) a solution is found in linear time.
- The trick: Create a graph, in which every node is an overlapping k-mer of the superstring and map edges to k-mers that have a) a prefix in the outgoing node and b) a suffix in the incoming node.

De Bruijn Graph Solution

- The trick: Create a graph, in which every node is an overlapping k-mer of the superstring and map edges to k-mers that have a)
  a prefix in the outgoing node and b) a suffix in the incoming node.



Hierholzer's algorithm

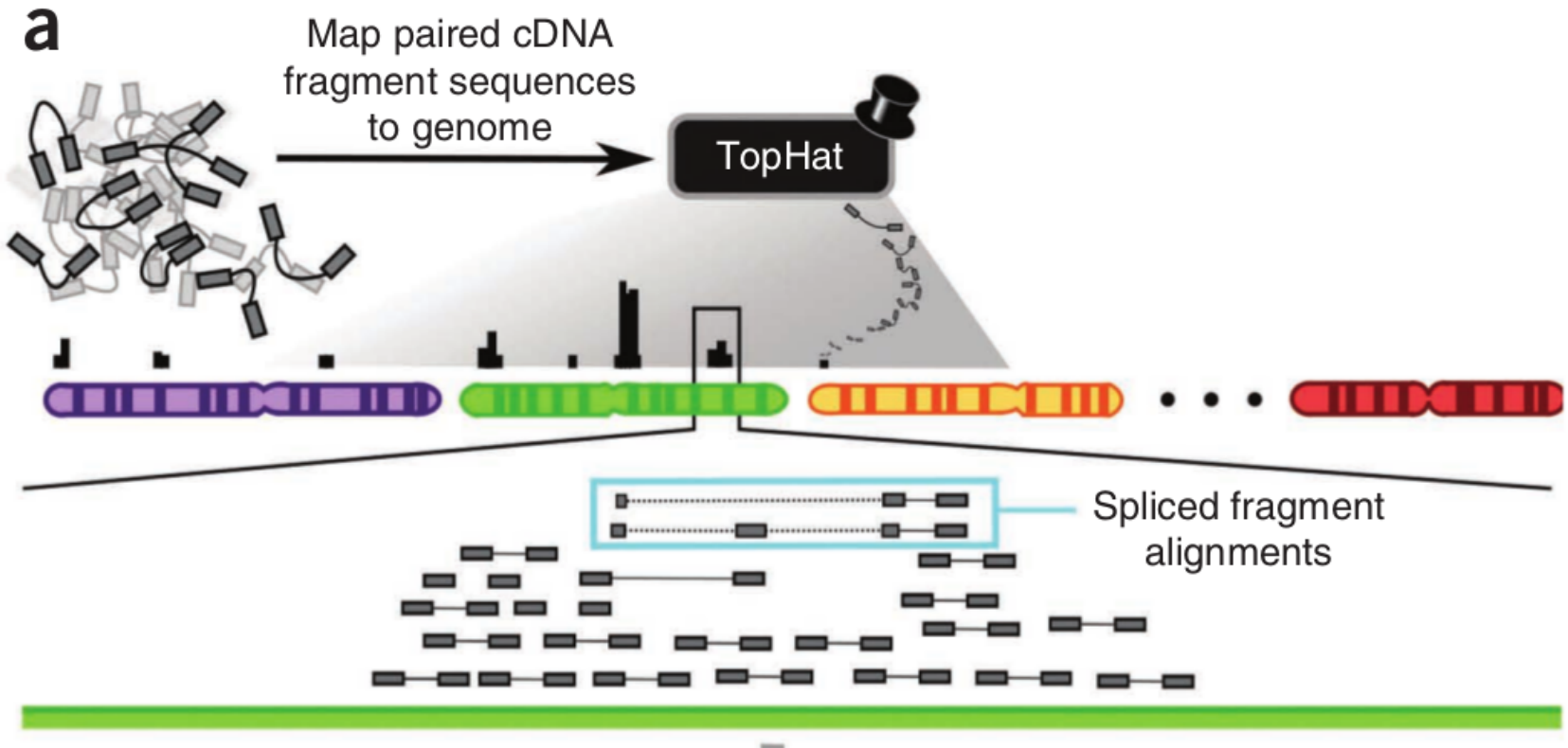
- Choose any starting vertex v, and follow a trail of edges from that vertex until returning to v. It is not possible to get stuck at any vertex other than v, because the even degree of all vertices ensures that, when the trail enters another vertex w there must be an unused edge leaving w. The tour formed in this way is a closed tour, but may not cover all the vertices and edges of the initial graph.
- As long as there exists a vertex u that belongs to the current tour but that has adjacent edges not part of the tour, start another trail from u, following unused edges until returning to u, and join the tour formed in this way to the previous tour.

2. Transcriptome Reconstruction

- The "easy" problem: Reads from an RNASeq experiment may result from spliced transcripts and thus need to be spliced-aligned to an assumed transcriptome
- The difficult problem: Reconstruct the transcriptome *de novo* by not assuming a known collection of transcripts
- The common problem: After reconstructing the transcriptome, performing *quantitative* assessments of each transcript at a given locus

The "easy" problem: Spliced Alignment

- Spliced-alignment is necessary when treating output that originates from mature transcripts (total RNA, or polyA+ RNA).
- Reads are broken into pieces and the algorithms (such as STAR, TopHat etc) try to map them in different regions of the genome that are a) either known exons from a pre-defined transcriptome or b) un-annotated.

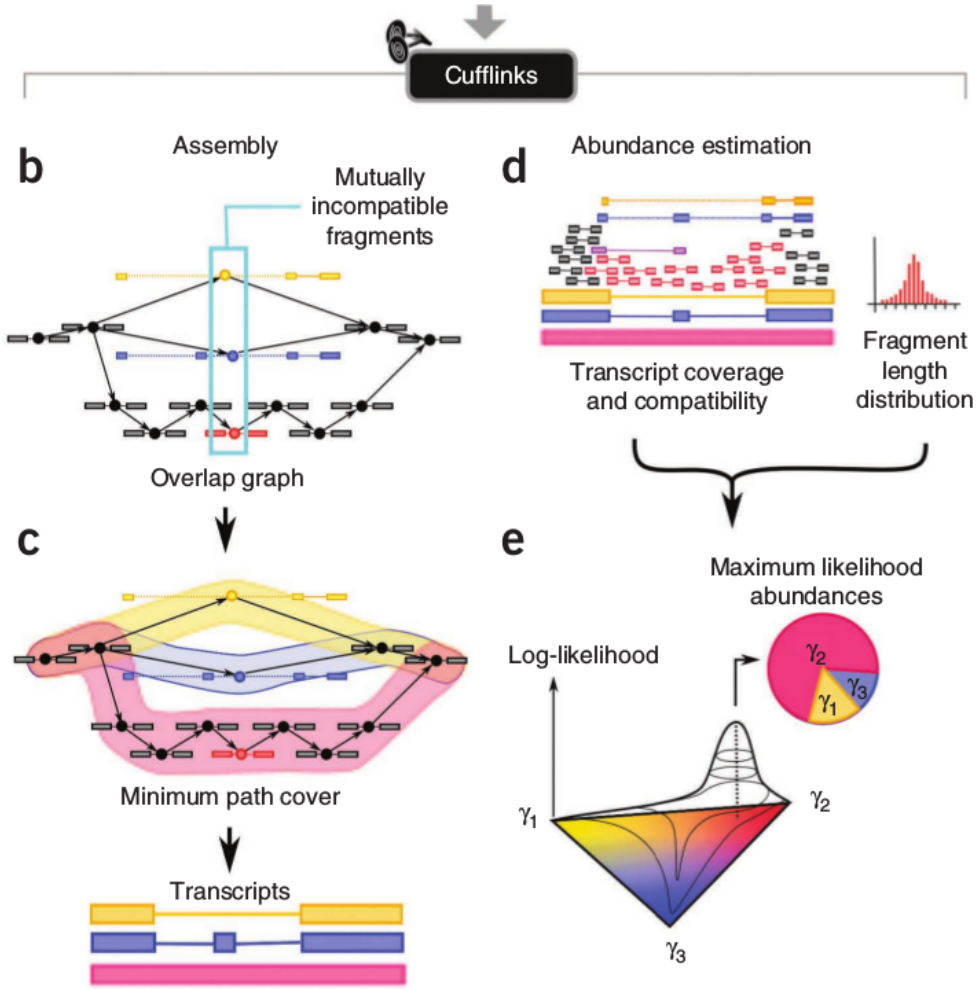


Approaches to Transcriptome Reconstruction

- The difficult problem of reconstructing the transcriptome results from the complexity of overlapping transcripts. Approaches that attempt to solve this problem include:
- Parsimony (Cufflinks)
- Statistical over-representation of spliced reads (Scripture)
- Optimized flow (StringTie)

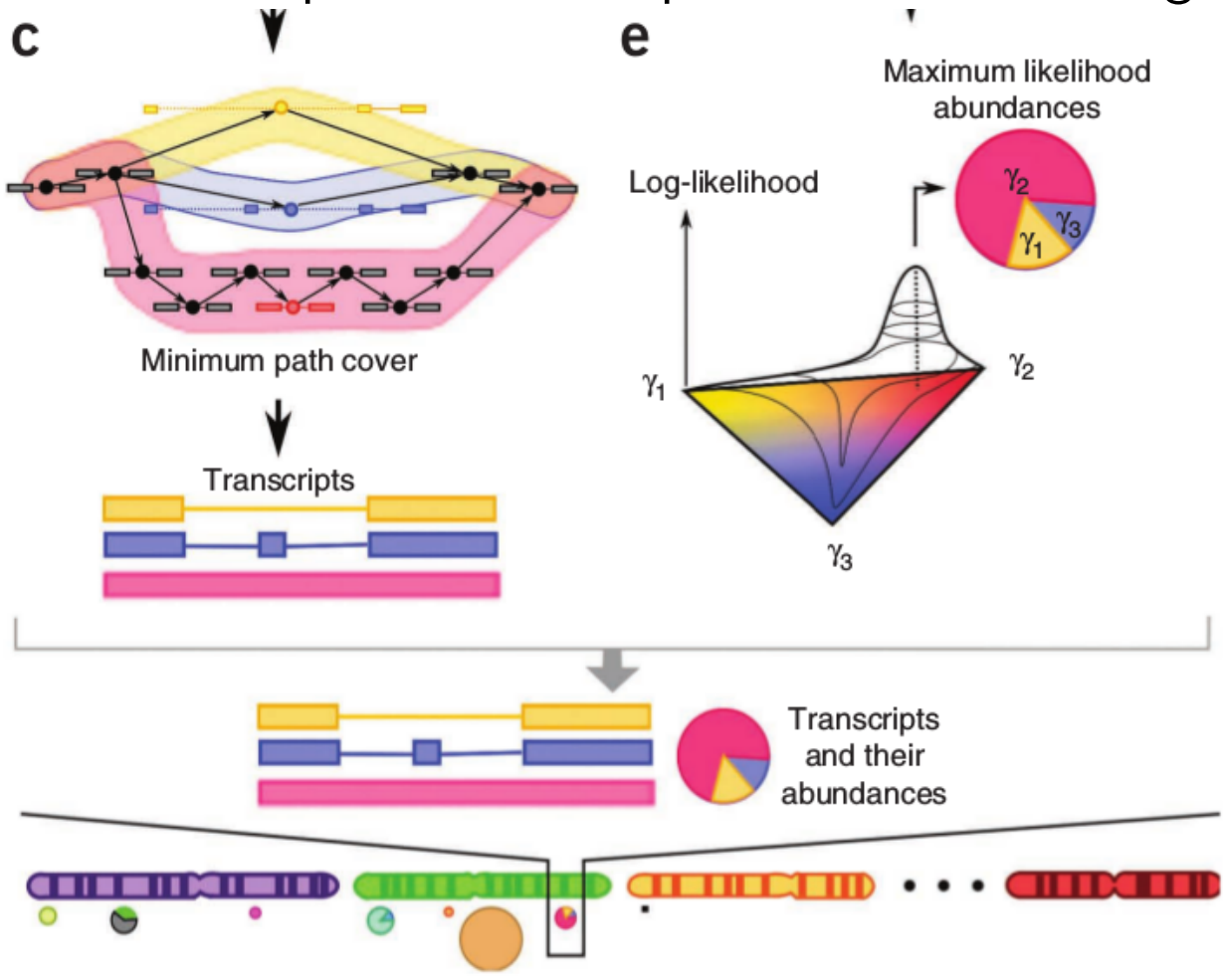
Parsimony criterion for Transcriptome Reconstruction

- Cufflinks (Trapnell et al, 2012), which is the most widely used transcript assembler, uses an overlap graph, in which the sequenced fragments are nodes and two nodes are connected if they overlap and have compatible splice patterns.



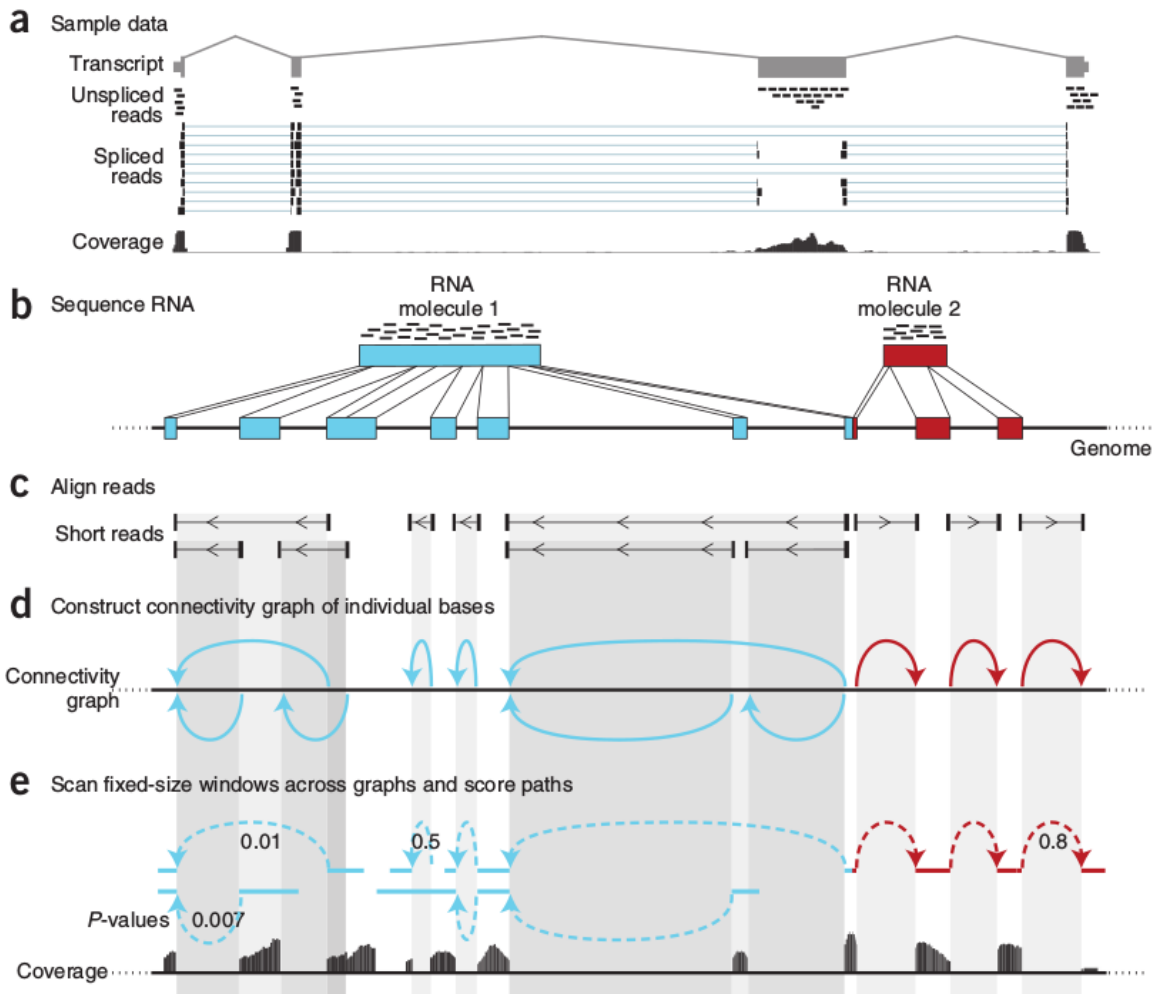
Parsimony criterion for Transcriptome Reconstruction

- Cufflinks uses a parsimony-based algorithm that generates the mini-
  mal number of transcripts that will explain all reads in the graph.



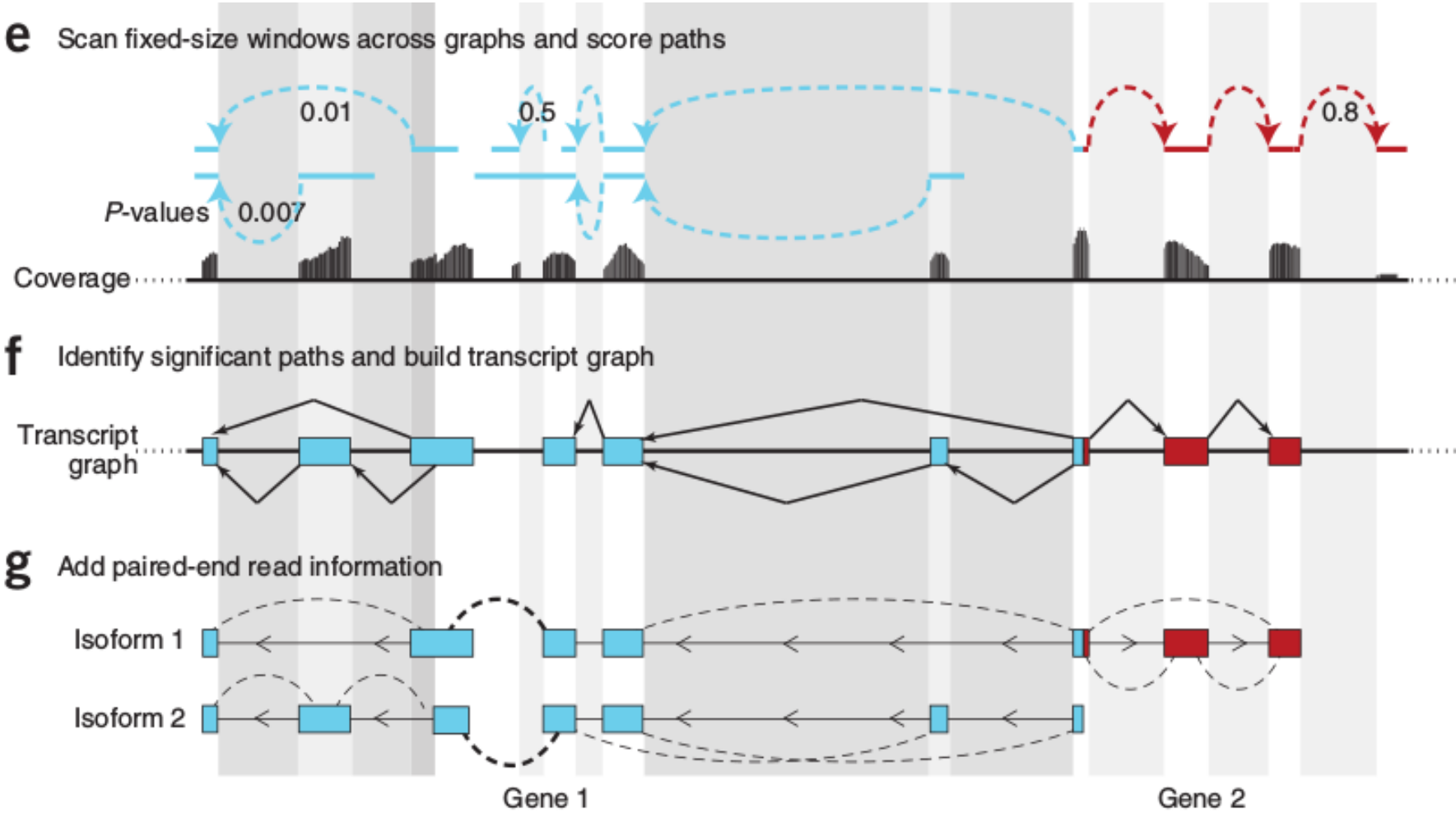
Statistical over-representation

- Scripture (Guttman et al, 2010) uses a statistical over-representation criterion to link exons based on the observed/expected ratio of spliced reads spanning the two regions. I



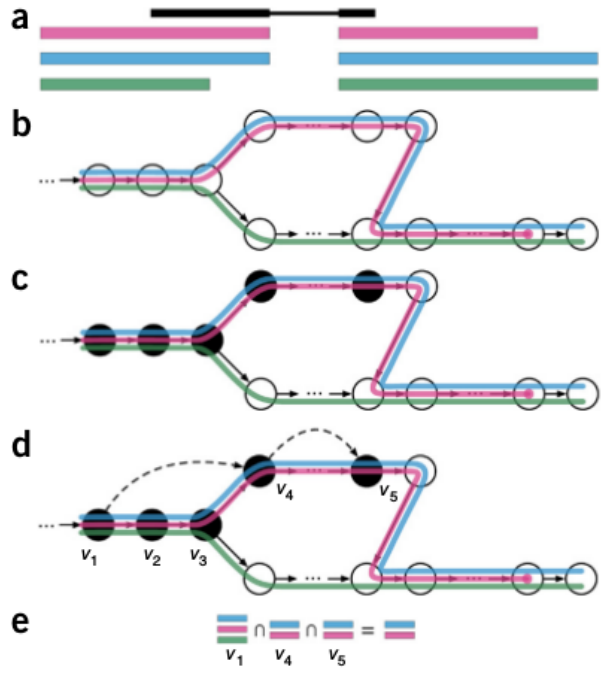
Statistical over-representation

- Scripture often leads to an over-estimation of alternative transcript isoforms



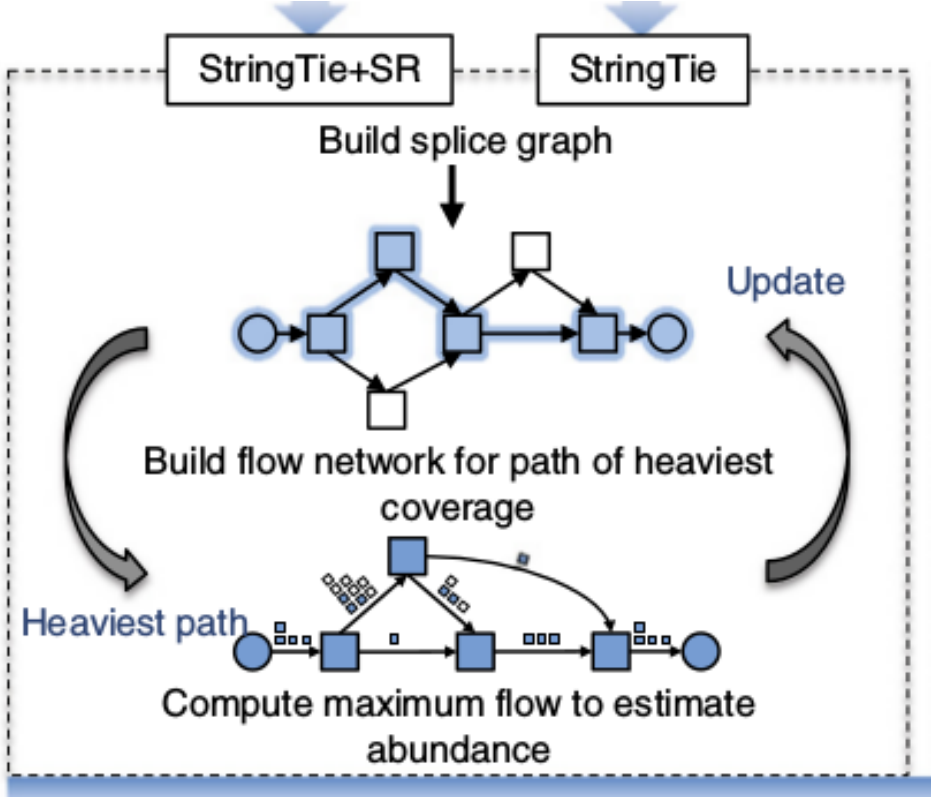
Other algorithms (flow, De Bruijn)

- Traph (Tomescu et al, 2013) solves a minimum cost flow optimization problem for transcript reconstruction. This is not exactly parsimony (but it is close).
- Kallisto (Bray et al., 2016) implements a de Bruijn Graph with the nodes being expected exons and attempts to solve the transcriptome reconstruction as the one with the highest "compatibility" with the reads. (a slightly ill-defined term)

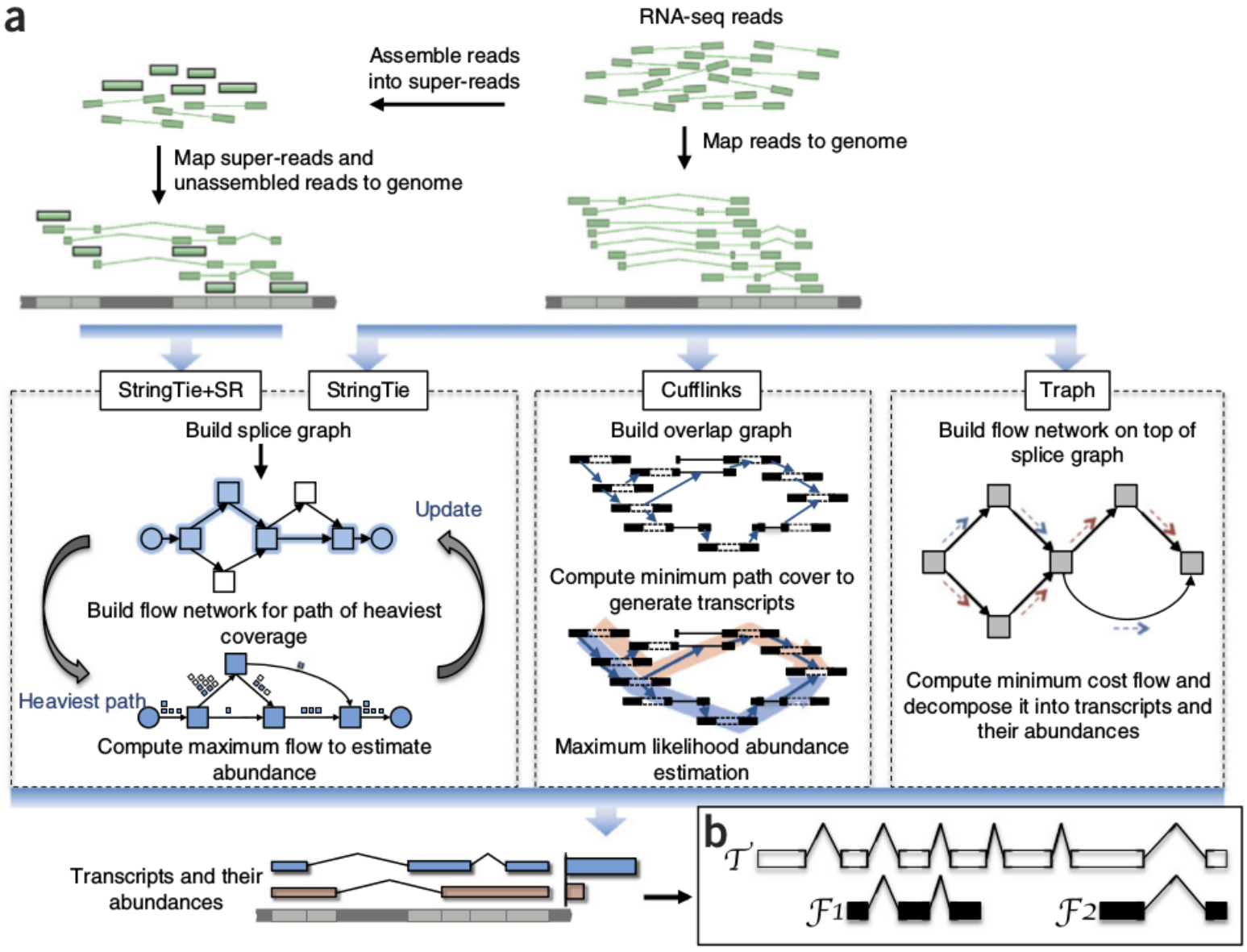


Maximum/Optimized Flow for Transcriptome Reconstruction

- StringTie assembles transcripts and estimates their expression levels simultaneously. StringTie *first groups the reads into clusters*, then creates a splice graph for each cluster from which it identifies transcripts, and then for each transcript it creates a separate flow network to estimate its expression level using *a maximum flow algorithm*.



Comparison of flow vs parsimony algorithms

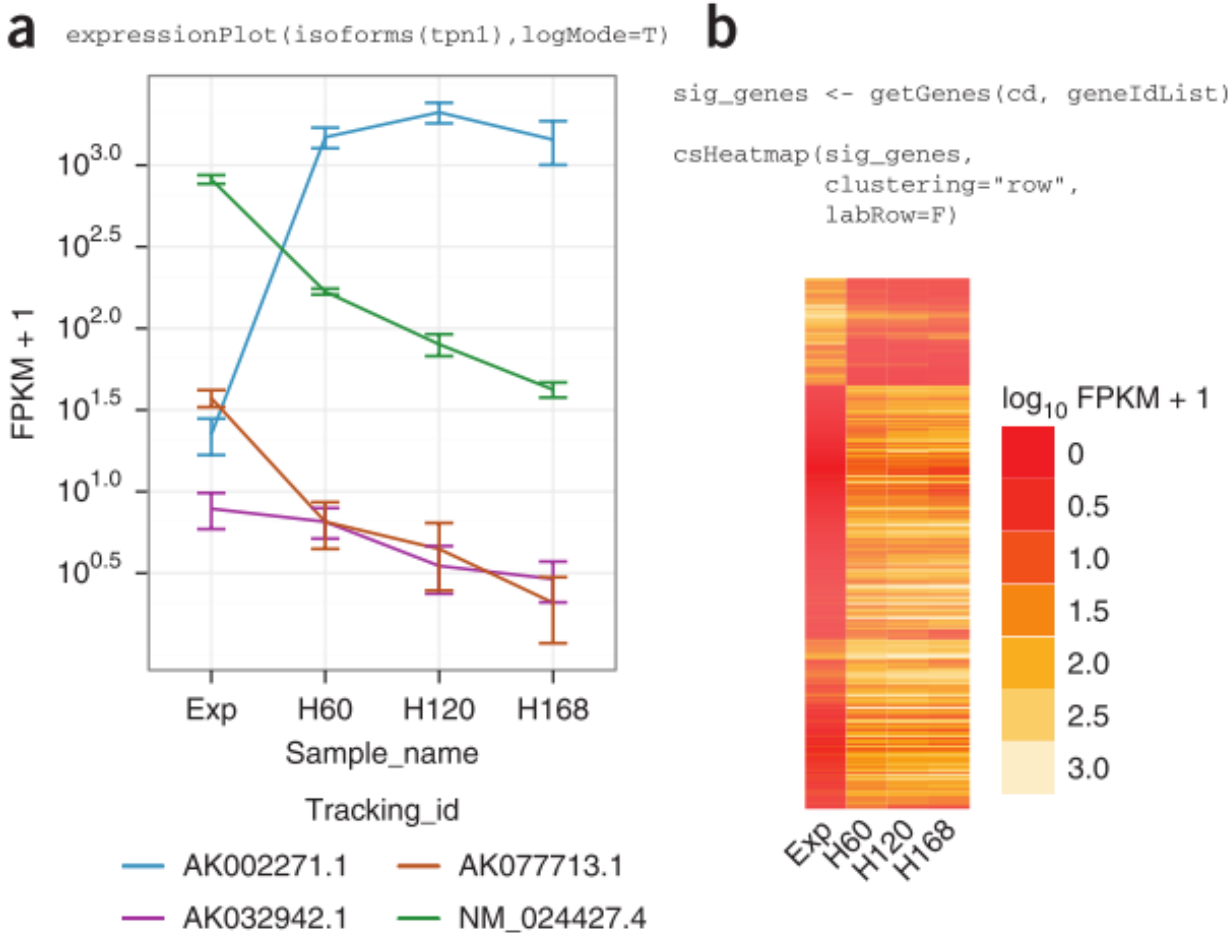


After the transcriptome reconstruction

- Merging of transcripts
- Transcriptome assembly works better **at low depths**. This is counter-intuitive but is related to the complexity of the problem.
- Calculation of FPKM values per transcript.
- Differential Expression assessment

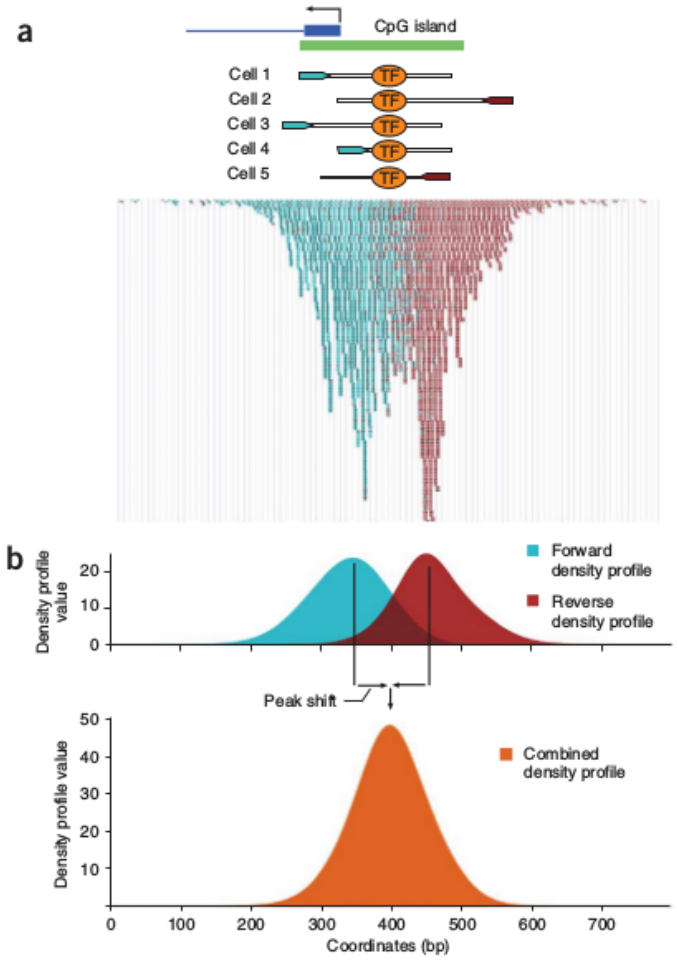
FPKM calculation/Differential Expression

- FPKM calculation:
  - Cufflinks. Performs a post-hoc statistical test once transcripts are defined
  - StringTie. Performs assessment in parallel with transcript reconstruction



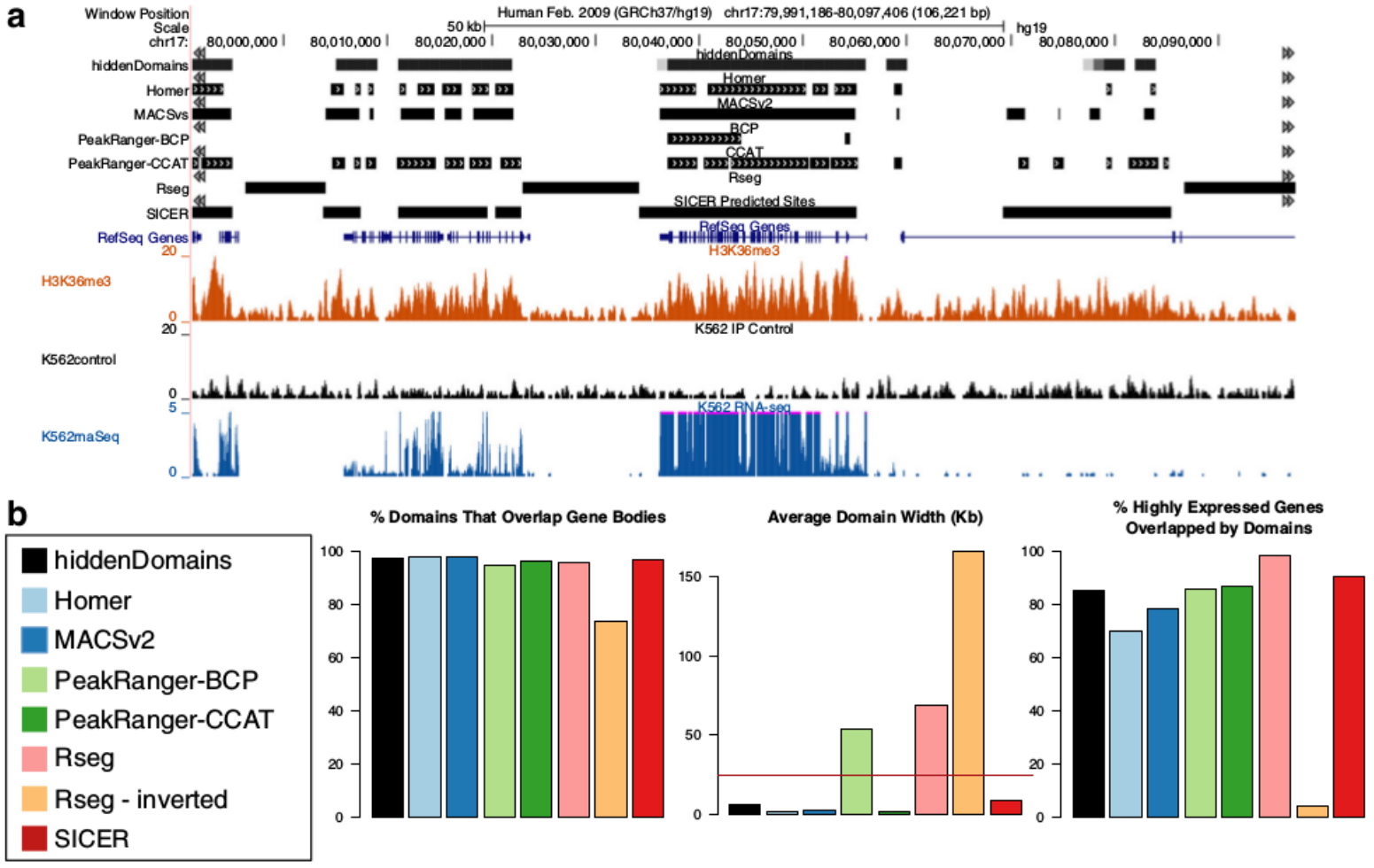
3. Peak Detection

- Most available methods (such as MACS or QUEST) make use of the directionality bias



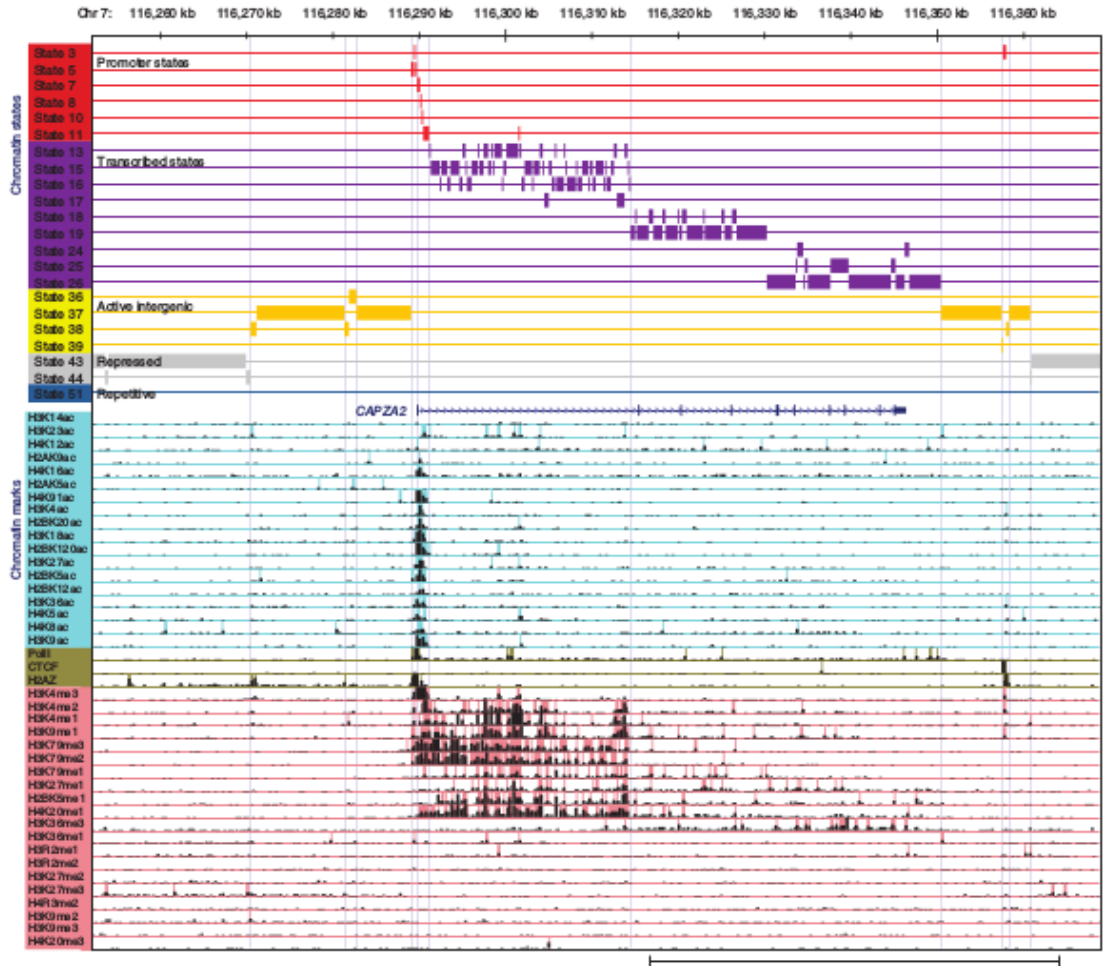
4. Domains of Epigenetic Marking

- Problems arise when enrichment of signal occurs in various different scales. Approaches that try to solve this problem use a variable window to define peaks at different length scales (e.g. Hidden Domains)



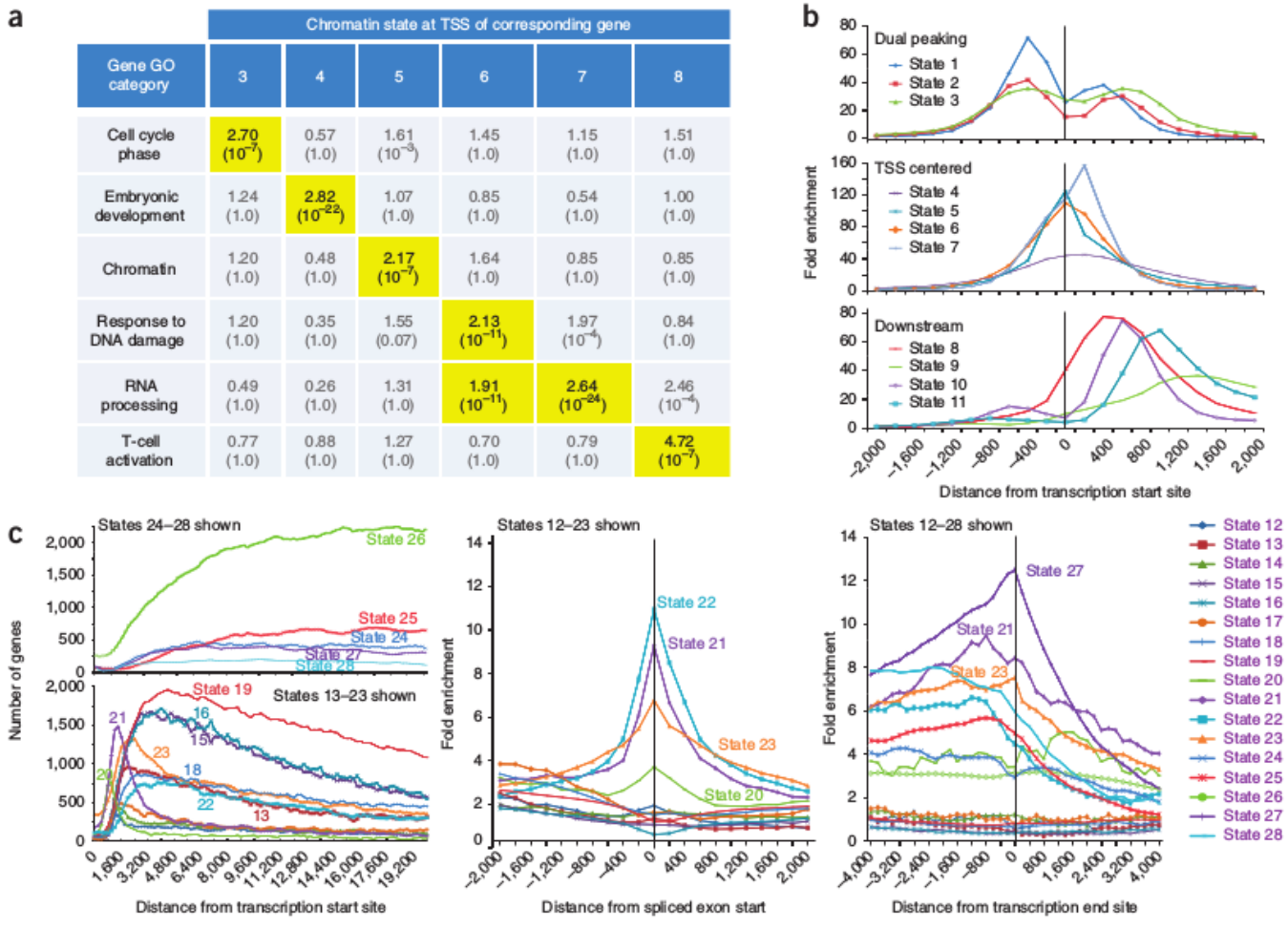
Chromatin States with HMMs

- One long-standing problem has been the segmentation of the genome in regions of consistent chromatin characteristics. Kellis and Ernst (2010) were the first to use Hidden Markov Models (HMMs) to segment the genome in regions depending on the combination of epigenetic marks.



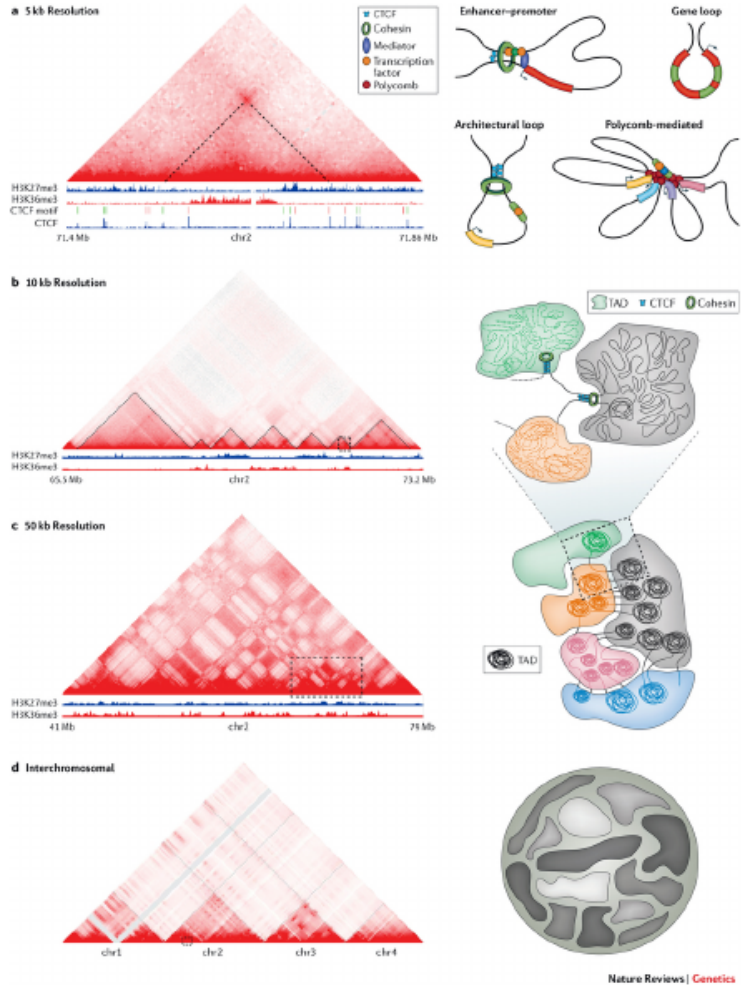
Chromatin States

- Chromatin States are linked to basic functional aspects of the genome, through a combination of a) genomic element comparisons and b) functional enrichment analyses



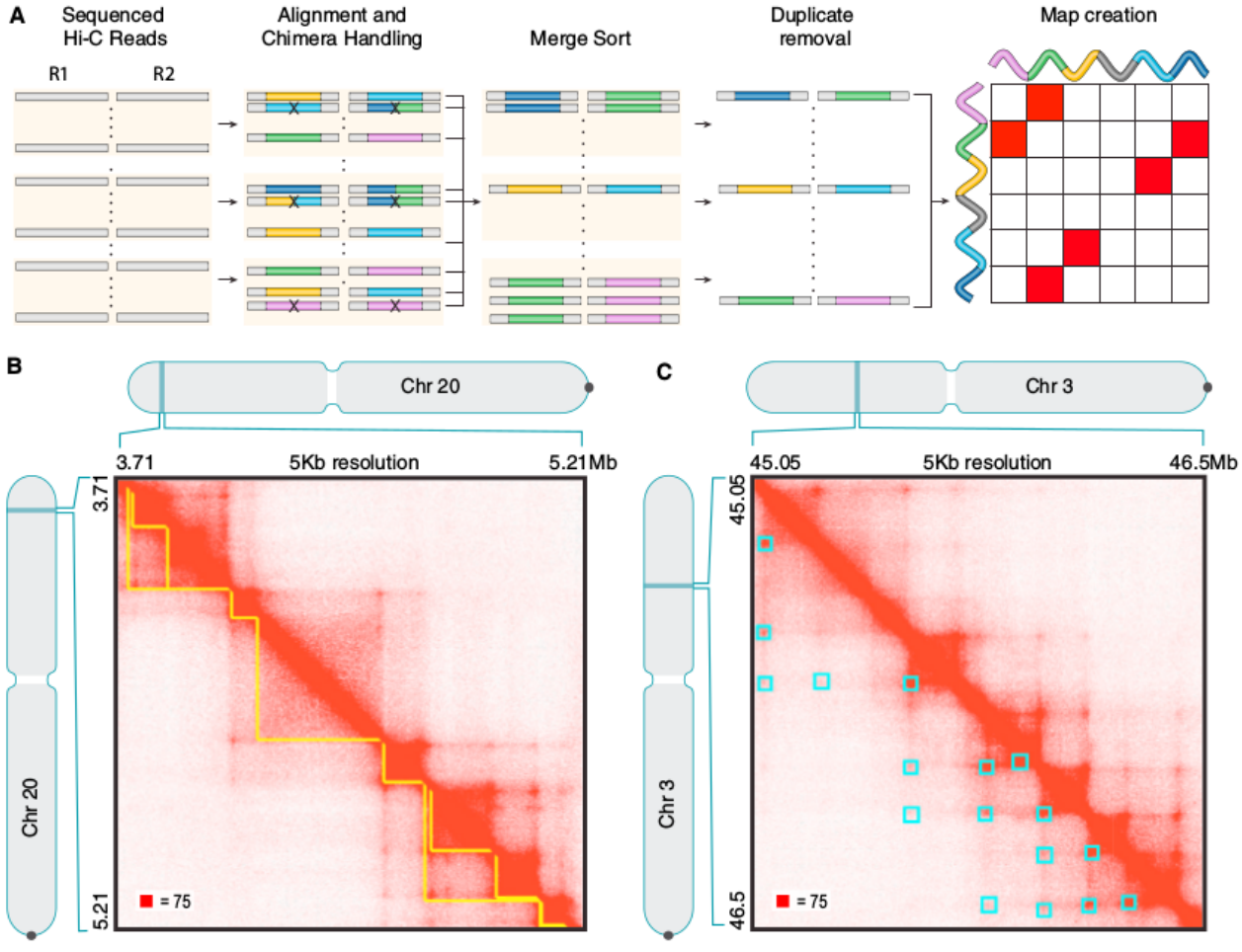
5. C-technology output analysis

- The goal of the C-technologies is to reconstruct the 3D structure of the genome by delineating areas of increased topological interactions, greater chromatin compartments etc.



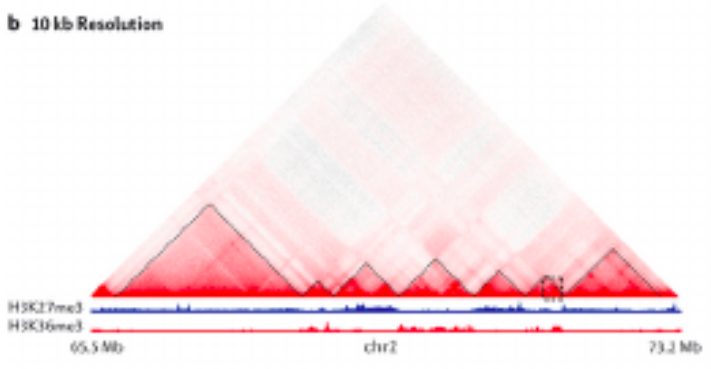
Analysis of interaction Maps

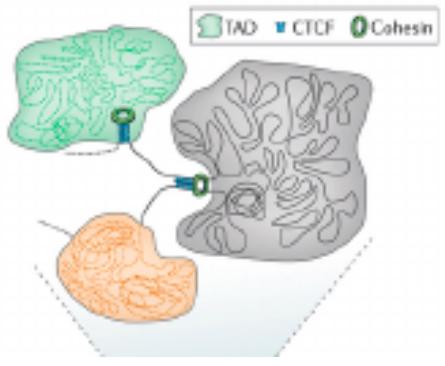
- Many techniques now exist for the analysis of reads from C-related outputs. The Juicer/Juicebox (Liebermann-Eiden lab) is the most commonly used



Topologically-Associated Domains (TAD)

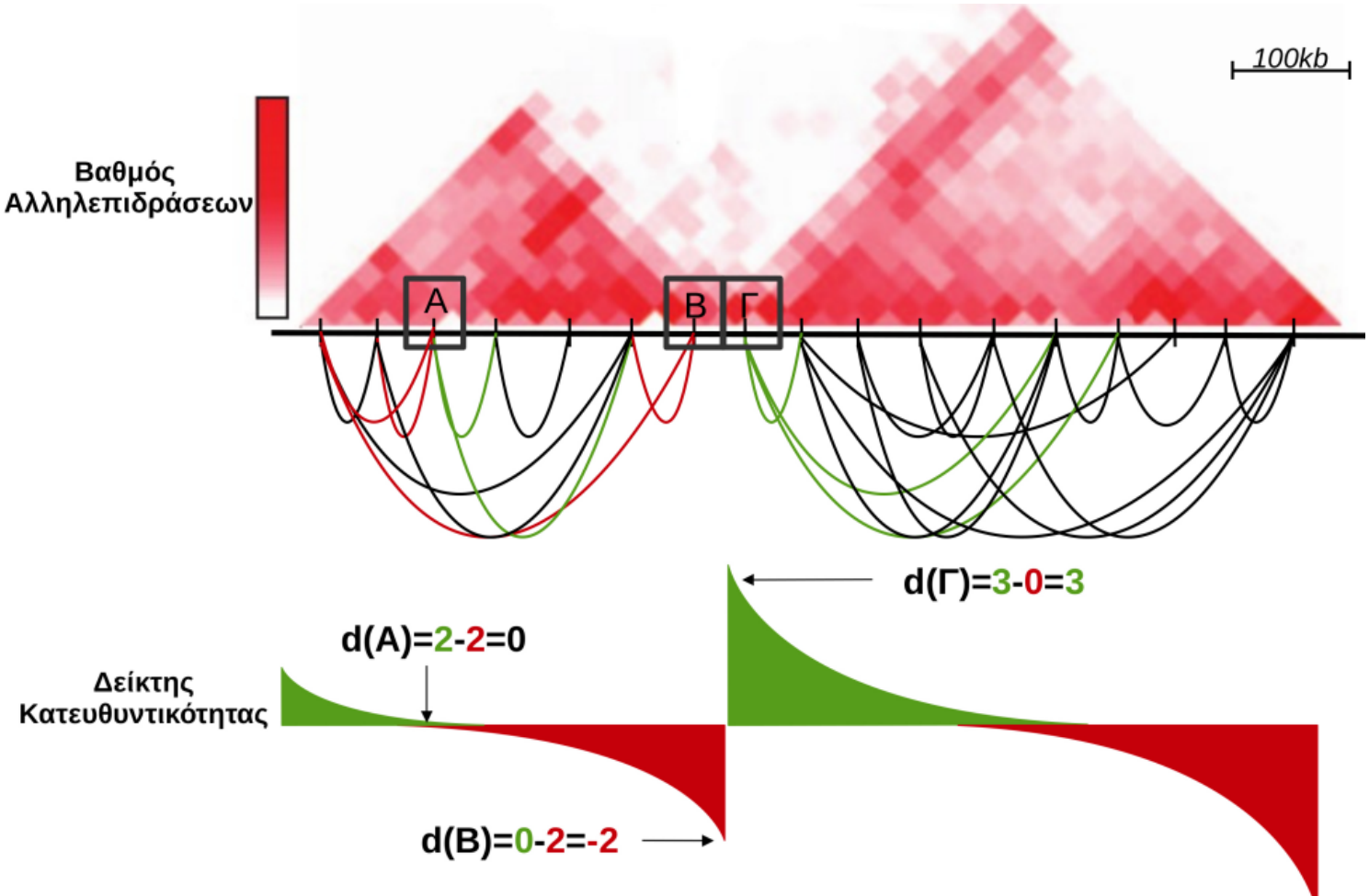
- They are the building blocks of chromatin structure as they are *invariant* to cell types and conditions
- They are defined as regions where increased interactions occur significantly more often **within** than **across** a linear chromosome





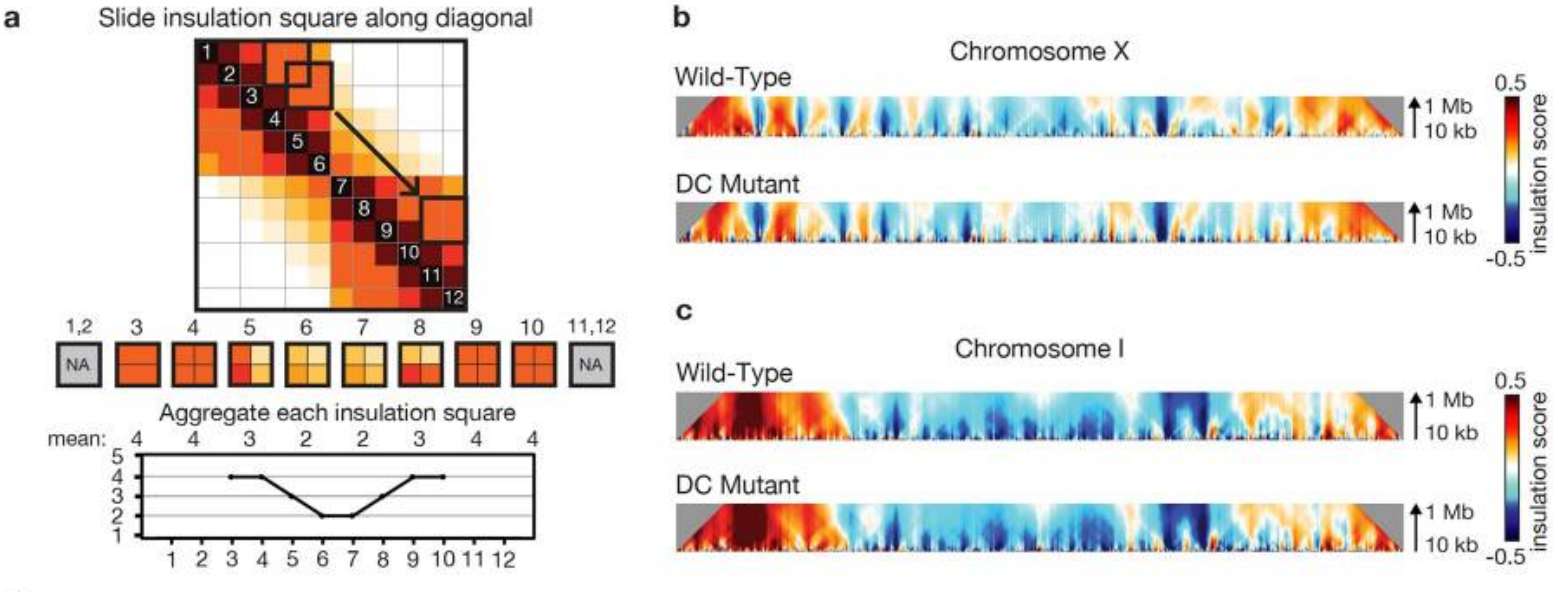
TAD definition: Directionality Index

- The DI calculates an algebraic sum of directed interactions. Each region gets a value that equals the sum of forward(+) and backward (-) interactions. Areas of the genome where a predominantly (-) sign, becomes predominantly (+) are representative of a TAD boundary



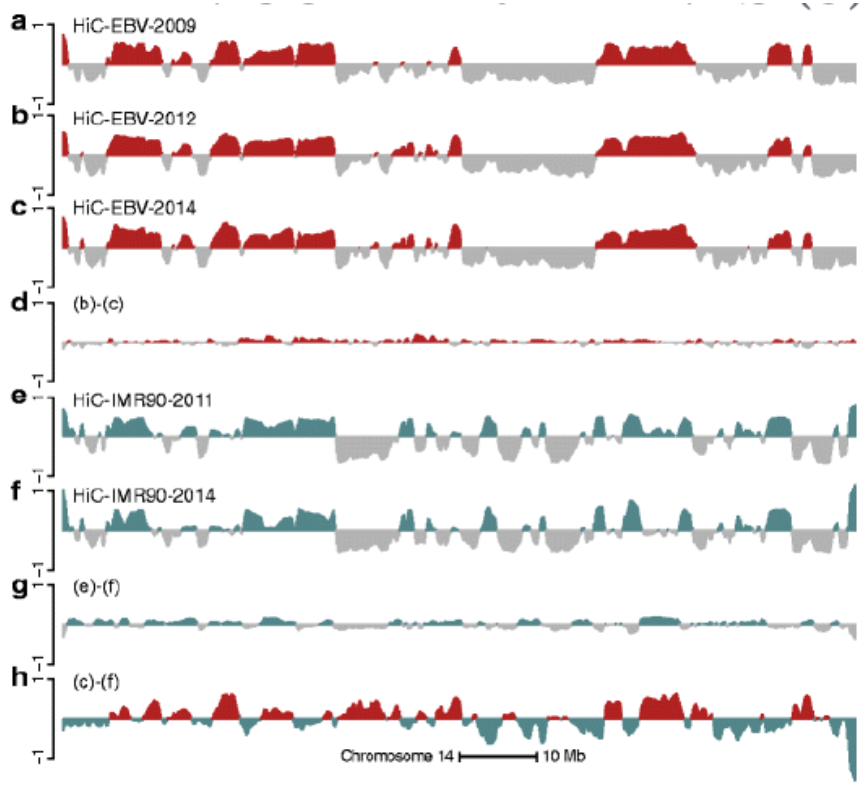
TAD definition: Insulation Profile

- The Insulation Profile takes fixed-sized squares of the interaction matrix (see Juicer output) and runs a sum across the diagonal. Local minima will be associated with insulating regions (that is very similar to TAD boundaries).



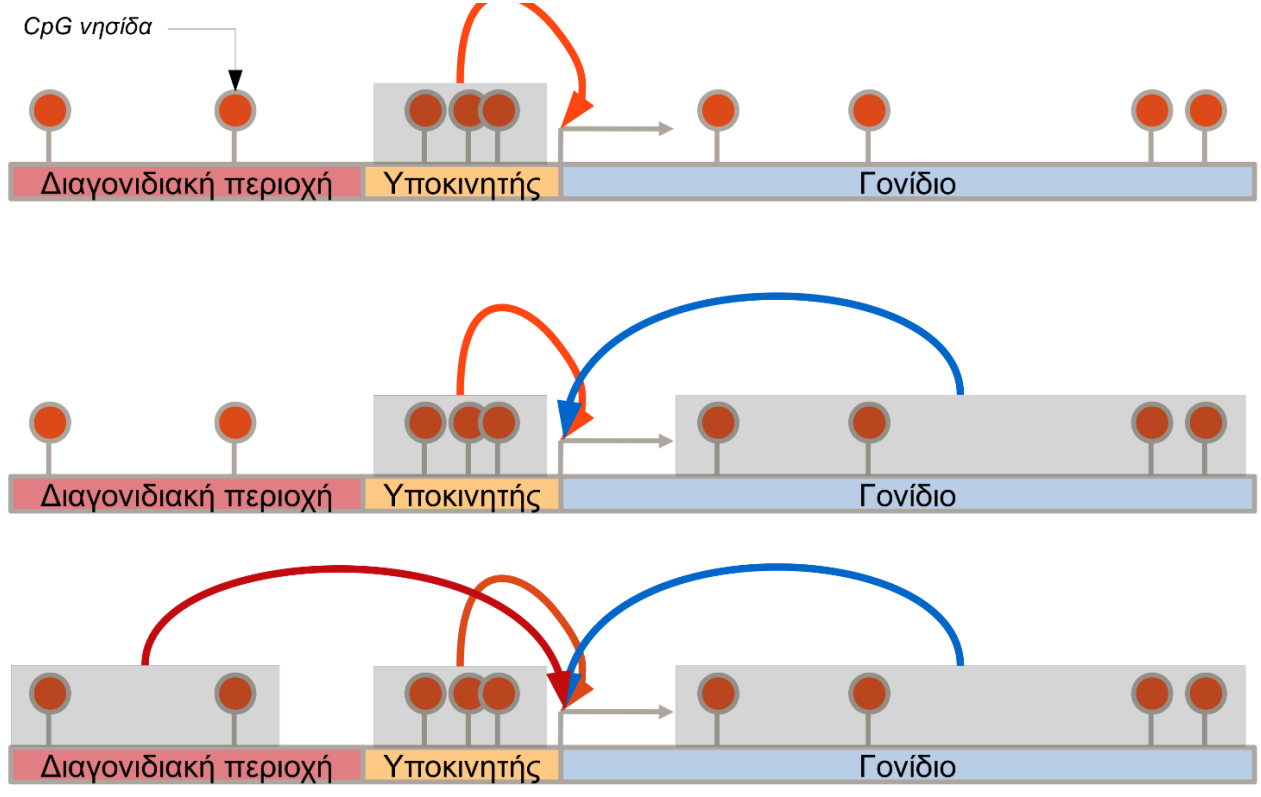
A/B Compartments

- A/B compartments are larger structures that are formed by the combination of many consecutive TADs. They are also reflected in the linear structure of the chromosome but are tissue-specific and condition-specific. They can be recreated through an Eigenvalue Decomposition of the original HiC interaction matrix.



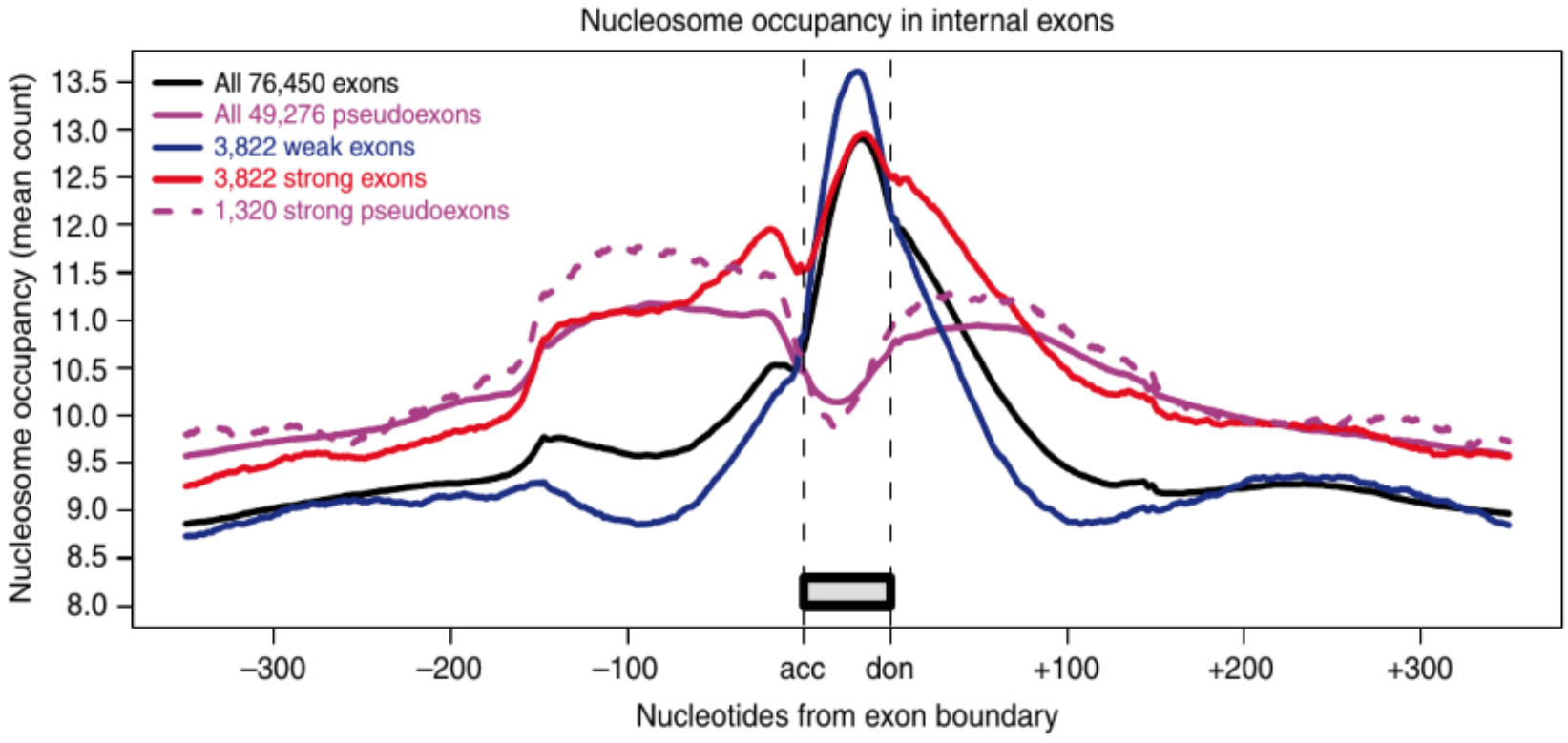
Further Analyses

- Comparison of coordinate files (e.g. two different peak files).
  - Assessment of overlap
  - Statistical significance of overlap
  - Location of closest-distance pairs



Further Analyses

- Link coordinate files with continuous-signal files
- Create average plots around fixed points (e.g. TSS)



Further Analyses

- Obtain values for heatmap/clustering