Graphs and Genomes

Michael Schatz

Bioinformatics Lecture 3
Quantitative Biology 2013

Dynamic Programming Matrix

Compute the optimal alignment of ABC…XY..N and DEF…UV…M

	0	A	B	C	…	X	Y	…	N
0									
D									
E									
F									
…									
U									
V									
…									
M									

Dynamic Programming Matrix

Compute the optimal alignment of ABC…XY..N and DEF…UV…M

	0	**A**	**B**	**C**	**…**	**X**	**Y**	**…**	**N**
0	0	1	2	3		X	X+1		N
D	1								
E	2								
F	3								
…									
U	U								
V	U+1								
…									
M	M								

Top row and first column are easy: it takes L-edits to transform and empty string into a length L string

Dynamic Programming Matrix

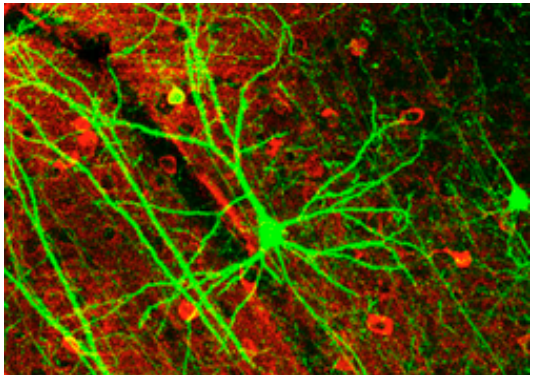
Compute the optimal alignment of "ABC…XY..N" and "DEF…UV…M"

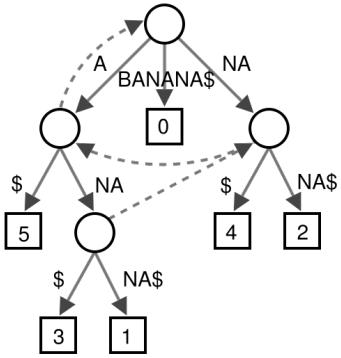
	0	**A**	**B**	**C**	**…**	**X**	**Y**	**…**	**N**
0	0	1	2	3		X	X+1		N
D	1								
E	2								
F	3								
…									
U	U					γ	α		
V	U+1					β	Ω		
…									
M	M								

$$\Omega = \min \begin{cases} \text{"Up"} + 1 & \alpha+1 \\ \text{"Left+} + 1 & \beta+1 \\ \text{"Diagonal"} +0/1 & \gamma+1 \end{cases}$$

Up	Left	Diagonal
ABC...XY-	ABC....XY	ABC...XY
DEF....UV	DEF...UV-	DEF...UV
α	β	γ

Biological Networks





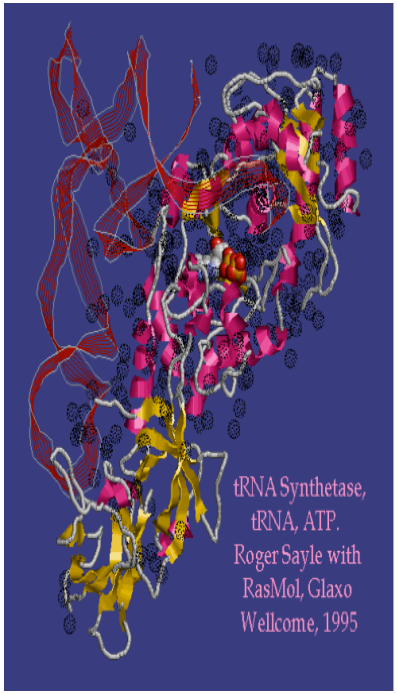
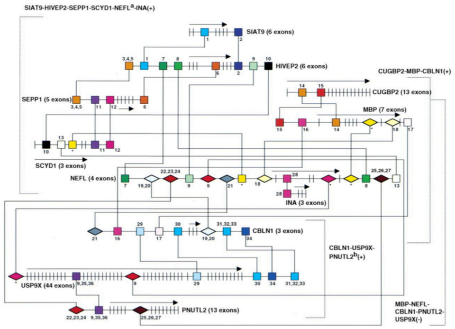
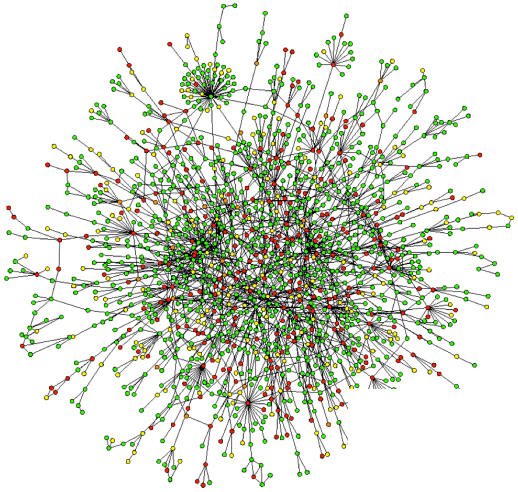


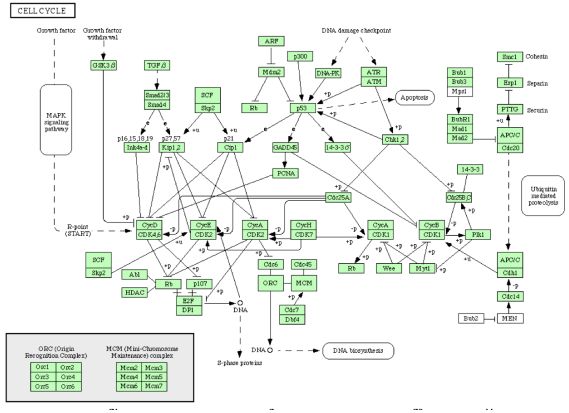
Figure 5 Putative regulatory elements shared between groups of correlated and anticorrelated genes

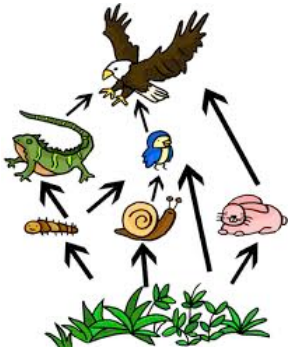


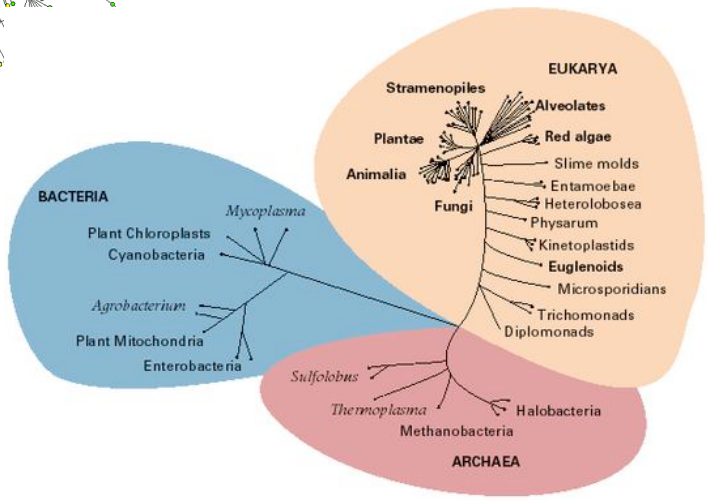
Vanessa M. Brown et al. Genome Res. 2002; 12: 868-884

Cold Spring Harbor Laboratory Press

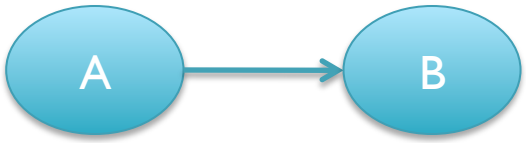








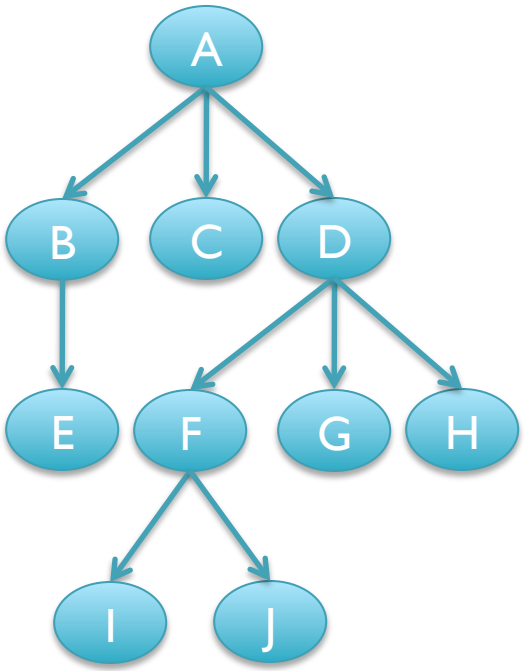
Graphs



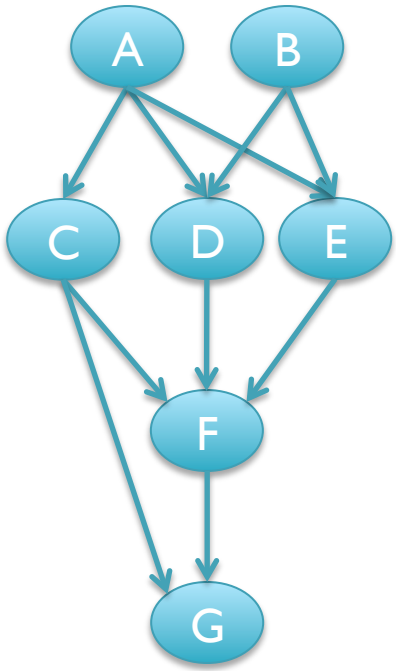
- Nodes
  - People, Proteins, Genes, Neurons, Sequences, Numbers, …
- Edges
  - A is connected to B
  - A is related to B
  - A regulates B
  - A precedes B
  - A interacts with B
  - A activates B
  - …

Graph Types

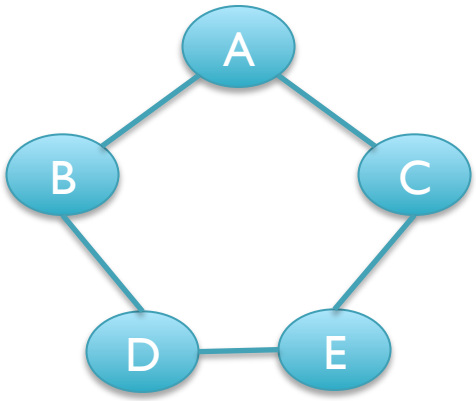
List



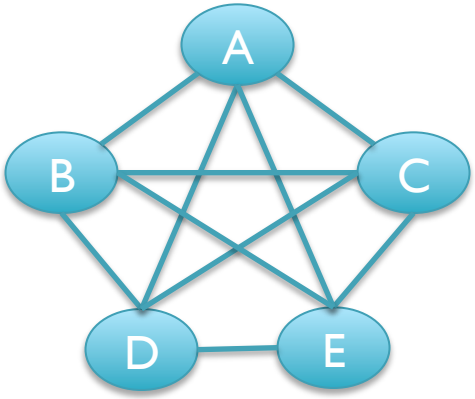
Tree



Directed
Acyclic
Graph

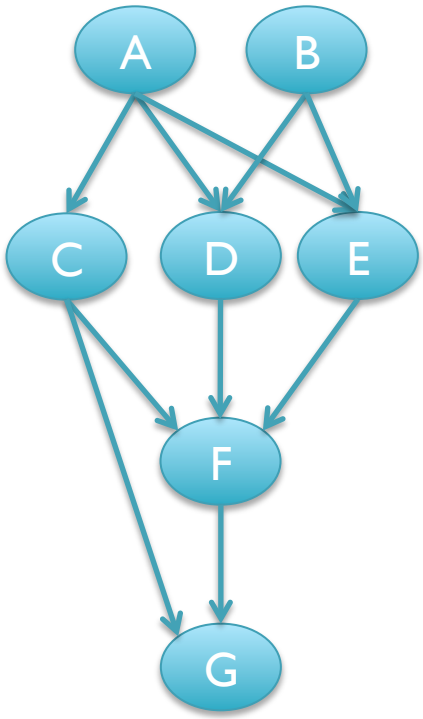


Cycle



Complete

Representing Graphs



Adjacency Matrix

Good for dense graphs
Fast, Fixed storage: N^2 bits

	A	B	C	D	E	F	G
A			1	1	1		
B				1	1		
C						1	1
D						1	
E						1	
F							1
G							

Adjacency List

Good for sparse graphs
Compact storage: 4 bytes/edge

A: C, D, E
B: D, E
C: F, G
D: F
E: F
G:

Edge List

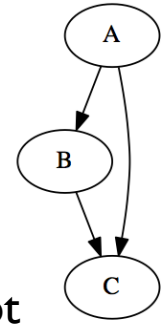
Easy, good if you (mostly) need to iterate through the edges
8 bytes / edge

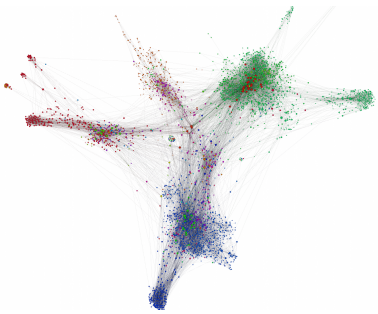
A,C B,C C,F
A,D B,D C,G
A,E B,E D,F
E,F F,G

Tools

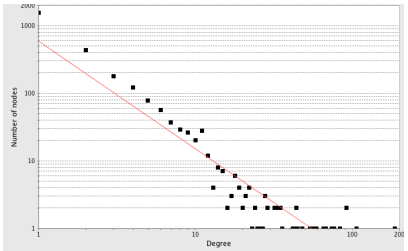
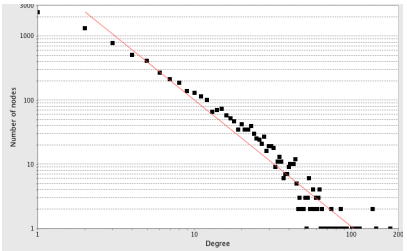
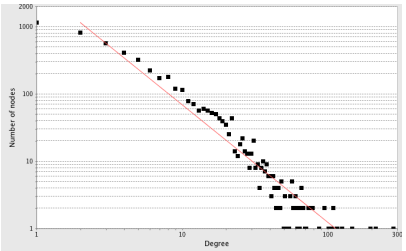
Matlab: http://www.mathworks.com/
Graphviz: http://www.graphviz.org/
Gephi: https://gephi.org/
Cytoscape: http://www.cytoscape.org/

```
digraph G {
    A->B
    B->C
    A->C
}
dot –Tpdf -og.pdf g.dot
```





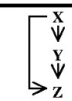
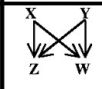
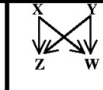
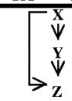
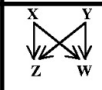
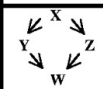
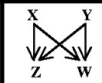
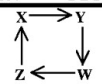
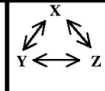
Network Characteristics

	C. elegans	*D. melanogaster*	*S. cerevisiae*
# Nodes	2646	7464	4965
# Edges	4037	22831	17536
Avg. / Max Degree	3.0 / 187	6.1 / 178	7.0 / 283
# Components	109	66	32
Largest Component	2386	7335	4906
Diameter	14	12	11
Avg. Shortest Path	4.8	4.4	4.1
Data Sources	2H	2x2H, TAP-MS	8x2H, 2xTAP, SUS
Degree Distributions			

Small World: Avg. Shortest Path between nodes is small
Scale Free: Power law distribution of degree – preferential attachment

Network Motifs

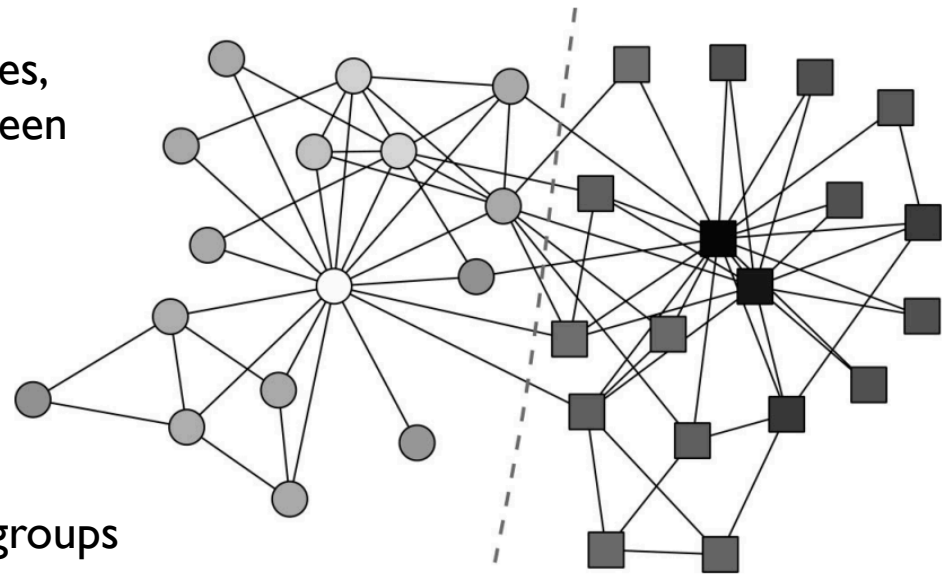
- Network Motif
  - Simple graph of connections
  - Exhaustively enumerate all possible 1, 2, 3, … k node motifs
- Statistical Significance
  - Compare frequency of a particular network motif in a real network as compared to a randomized network
- Certain motifs are "characteristic features" of the network

Network	Nodes	Edges	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score	N_{real}	$N_{rand} \pm SD$	Z score
Gene regulation (transcription)					**Feed-forward loop**			**Bi-fan**			
E. coli	424	519	40	7 ± 3	10	203	47 ± 12	13			
*S. cerevisiae**	685	1,052	70	11 ± 4	14	1812	300 ± 40	41			
Neurons					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
C. elegans†	252	509	125	90 ± 10	3.7	127	55 ± 13	5.3	227	35 ± 10	20
Food webs					**Three chain**			**Bi-parallel**			
Little Rock	92	984	3219	3120 ± 50	2.1	7295	2220 ± 210	25			
Ythan	83	391	1182	1020 ± 20	7.2	1357	230 ± 50	23			
St. Martin	42	205	469	450 ± 10	NS	382	130 ± 20	12			
Chesapeake	31	67	80	82 ± 4	NS	26	5 ± 2	8			
Coachella	29	243	279	235 ± 12	3.6	181	80 ± 20	5			
Skipwith	25	189	184	150 ± 7	5.5	397	80 ± 25	13			
B. Brook	25	104	181	130 ± 7	7.4	267	30 ± 7	32			
Electronic circuits (forward logic chips)					**Feed-forward loop**			**Bi-fan**			**Bi-parallel**
s15850	10,383	14,240	424	2 ± 2	285	1040	1 ± 1	1200	480	2 ± 1	335
s38584	20,717	34,204	413	10 ± 3	120	1739	6 ± 2	800	711	9 ± 2	320
s38417	23,843	33,661	612	3 ± 2	400	2404	1 ± 1	2550	531	2 ± 2	340
s9234	5,844	8,197	211	2 ± 1	140	754	1 ± 1	1050	209	1 ± 1	200
s13207	8,651	11,831	403	2 ± 1	225	4445	1 ± 1	4950	264	2 ± 1	200
Electronic circuits (digital fractional multipliers)					**Three-node feedback loop**			**Bi-fan**			**Four-node feedback loop**
s208	122	189	10	1 ± 1	9	4	1 ± 1	3.8	5	1 ± 1	5
s420	252	399	20	1 ± 1	18	10	1 ± 1	10	11	1 ± 1	11
s838‡	512	819	40	1 ± 1	38	22	1 ± 1	20	23	1 ± 1	25
World Wide Web					**Feedback with two mutual dyads**			**Fully connected triad**			**Uplinked mutual dyad**
nd.edu§	325,729	1.46e6	1.1e5	2e3 ± 1e2	800	6.8e6	5e4±4e2	15,000	1.2e6	1e4 ± 2e2	5000

Network Motifs: Simple Building Blocks of Complex Networks
Milo et al (2002) *Science. 298:824-827*

Modularity

- Community structure
  - Densely connected groups of vertices, with only sparser connections between groups
  - Reveals the structure of large-scale network data sets



- Modularity
  - The number of edges falling within groups minus the expected number in an equivalent network with edges placed at random
  - Larger positive values => Stronger community structure
  - Optimal assignment determined by computing the eigenvector of the modularity matrix

$$Q = \boxed{\frac{1}{4m}} \sum_{ij} \left(\boxed{A_{ij}} - \boxed{\frac{k_i k_j}{2m}} \right) \boxed{(s_i s_j + 1)}$$

Normalization factor

Adjacency matrix

Random Prob. (product of degrees)

Indicates same group

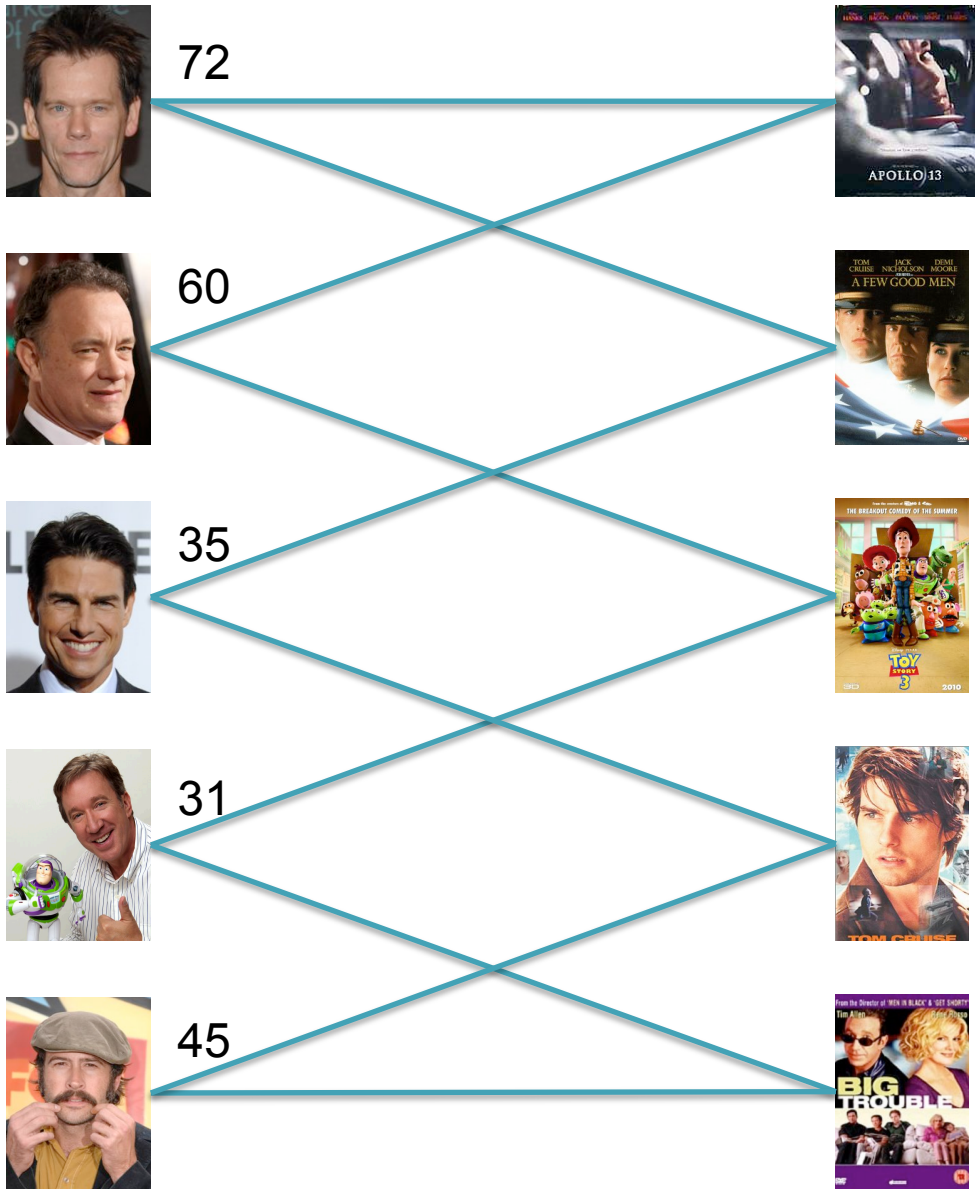
Modularity and community structure in networks.
Newman ME (2006) *PNAS. 103(23) 8577-8582*

Kevin Bacon and Bipartite Graphs

Find the ***shortest*** path from Kevin Bacon to Jason Lee

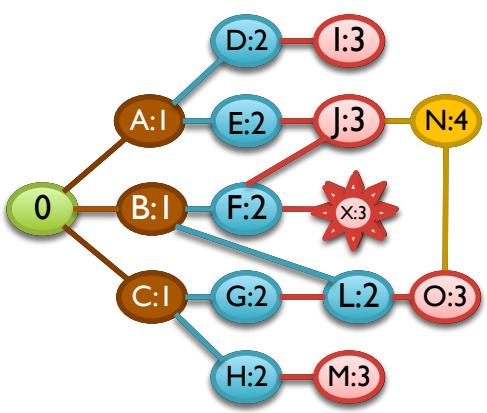
Breadth First Search: 4 hops

Bacon Distance: 2



BFS

```
BFS(start, stop)
// initialize all nodes dist = -1
start.dist = 0
list.addEnd(start)
while (!list.empty())
   cur = list.begin()
   if (cur == stop)
      print cur.dist;
   else
      foreach child in cur.children
         if (child.dist == -1)
            child.dist = cur.dist+1
            list.addEnd(child)
```



0

A,B,C
B,C,D,E
C,D,E,F,L

D,E,F,L,G,H
E,F,L,G,H,I
F,L,G,H,I,J
L,G,H,I,J,X
G,H,I,J,X,O
H,I,J,X,O

I,J,X,O,M
J,X,O,M
X,O,M,N
O,M,N
M,N

N

[How many nodes will it visit?]

[What's the running time?]

[What happens for disconnected components?]

BFS

```
BFS(start, stop)
// initialize all nodes dist = -1
start.dist = 0
list.addEnd(start)
while (!list.empty())
   cur = list.begin()
   if (cur == stop)
      print cur.dist;
   else
      foreach child in cur.children
         if (child.dist == -1)
            child.dist = cur.dist+1
            list.addEnd(child)
```

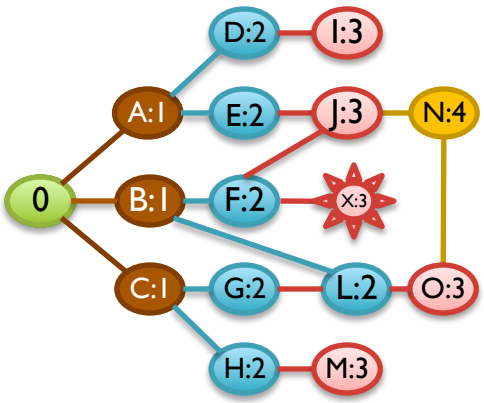
0

A,B,C
B,C,D,E
C,D,E,F,L

D,E,F,L,G,H
E,F,L,G,H,I
F,L,G,H,I,J
L,G,H,I,J,X
G,H,I,J,X,O
H,I,J,X,O

I,J,X,O,M
J,X,O,M
X,O,M,N
O,M,N
M,N

N



DFS

```
DFS(start, stop)
// initialize all nodes dist = -1
start.dist = 0
list.addEnd(start)
while (!list.empty())
   cur = list.end()
   if (cur == stop)
      print cur.dist;
   else
      foreach child in cur.children
         if (child.dist == -1)
            child.dist = cur.dist+1
            list.addEnd(child)
```

0

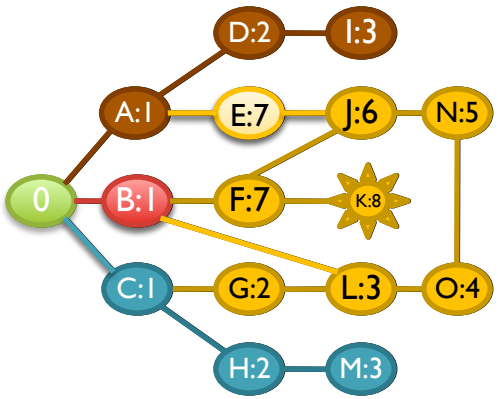
A,B,C

A,B,G,H
A,B,G,M

A,B,G
A,B,L
A,B,O
A,B,N
A,B,J
A,B,E,F
A,B,E,K
A,B,E

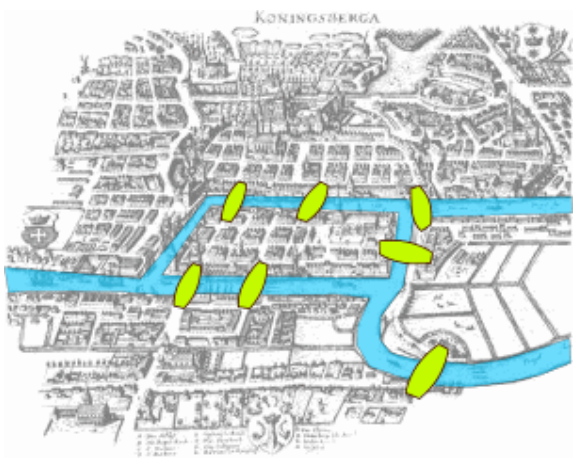
A,B

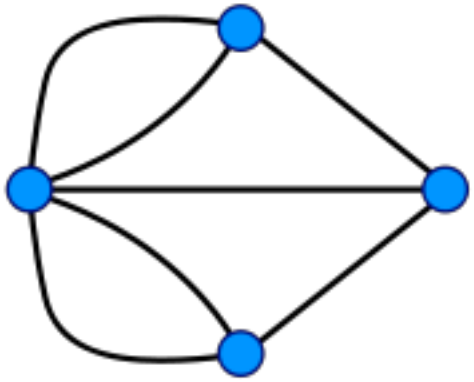
A
D
I



Eulerian Cycle Problem

- **Seven Bridges of Königsberg**
  - Find a cycle that visits every ***edge*** exactly once





[Can you find the cycle?]

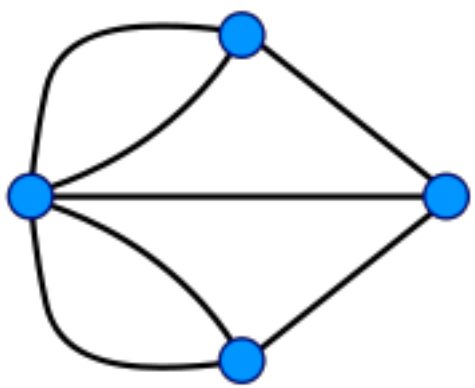
bioalgorithms.info

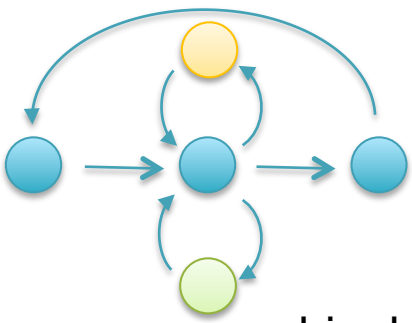
Euler Theorem

- A graph is ***balanced*** if for every vertex the number of incoming edges equals to the number of outgoing edges:

$$in(v)=out(v)$$

- **Theorem**: *A connected graph is Eulerian if and only if each of its vertices is balanced.*

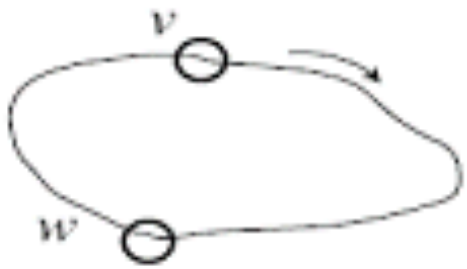




bioalgorithms.info

Algorithm for Constructing an Eulerian Cycle

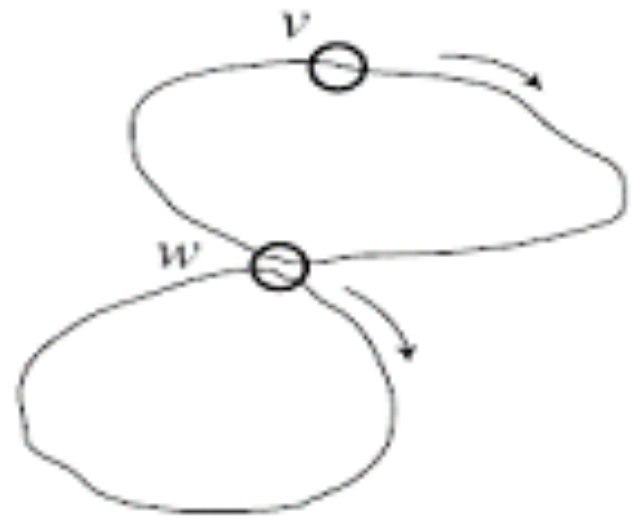
a. Start with an arbitrary vertex *v* and form an arbitrary cycle with unused edges until a dead end is reached. Since the graph is Eulerian this dead end is necessarily the starting point, i.e., vertex *v*.



(a)

Algorithm for Constructing an Eulerian Cycle (cont'd)

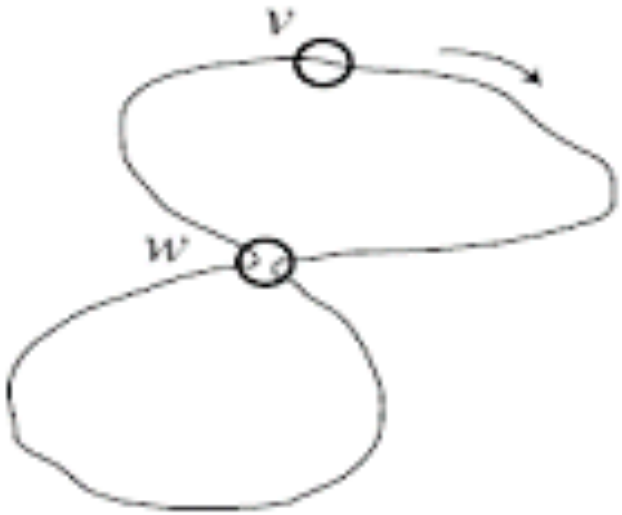
b. If cycle from (a) above is not an Eulerian cycle, it must contain a vertex *w*, which has untraversed edges. Perform step (a) again, using vertex *w* as the starting point. Once again, we will end up in the starting vertex *w*.



(b)

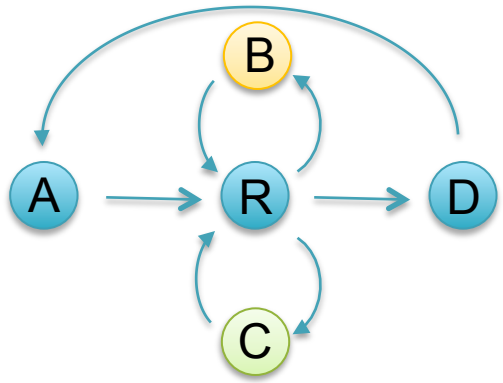
Algorithm for Constructing an Eulerian Cycle (cont'd)

c. Combine the cycles from (a) and (b) into a single cycle and iterate step (b).



(c)

Counting Eulerian Cycles



ARBRCRD
or
ARCRBRD

Generally an exponential number of compatible sequences

– Value computed by application of the BEST theorem (Hutchinson, 1975)

$$\mathcal{W}(G,t) = (\det L)\left\{\prod_{u\in V}(r_u - 1)!\right\}\left\{\prod_{(u,v)\in E} a_{uv}!\right\}^{-1}$$

L = *n x n* matrix with r_u-a_{uu} along the diagonal and -a_{uv} in entry uv

$r_u = d^+(u)+1$ if $u=t$, or $d^+(u)$ otherwise

a_{uv} = multiplicity of edge from *u* to *v*

Assembly Complexity of Prokaryotic Genomes using Short Reads.
Kingsford C, Schatz MC, Pop M (2010) *BMC Bioinformatics*.

BFS and TSP

- BFS computes the shortest path between a pair of nodes in $O(|E|) = O(|N|^2)$

- What if we wanted to compute the shortest path visiting every node once?
  - Traveling Salesman Problem

ABDCA: 4+2+5+3 = 14

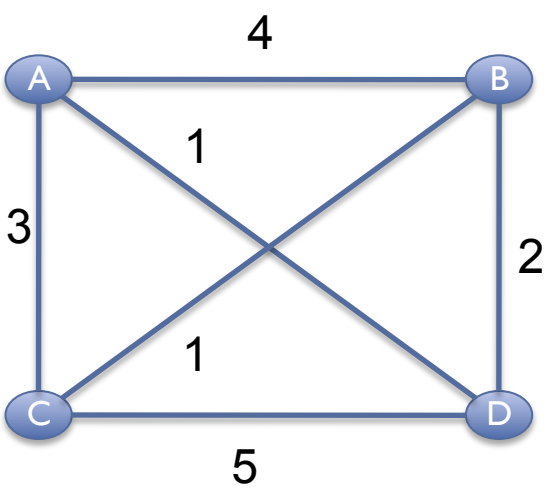
ACDBA: 3+5+2+4 = 14*

ABCDA: 4+1+5+1 = 11

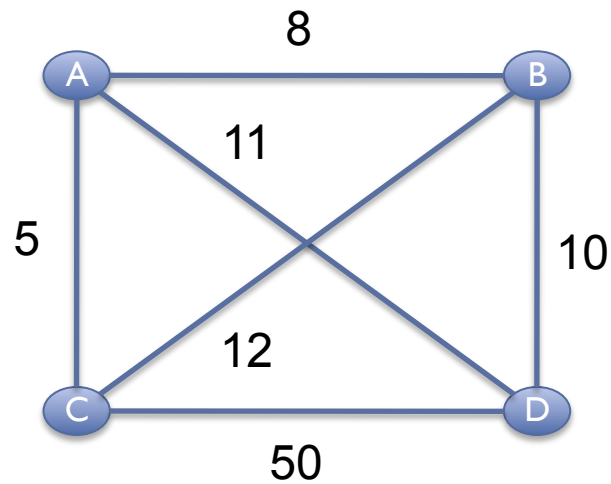
ADCBA: 1+5+1+4 = 11*

ACBDA: 3+1+2+1 = 7

ADBCA: 1+2+1+3= 7 *



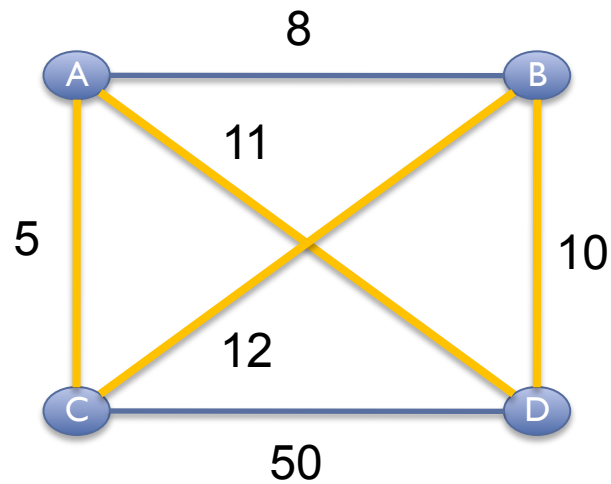
Greedy Search



Greedy Search

Greedy Search

```
cur=graph.randNode()
while (!done)
    next=cur.getNextClosest()
```



Greedy: ABDCA = 5+8+10+50= 73

Optimal: ACBDA = 5+11+10+12 = 38

Greedy finds the global optimum only when

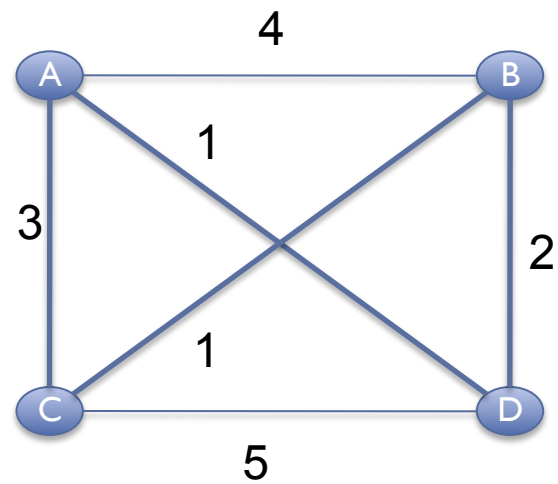
1. Greedy Choice: Local is correct without reconsideration
2. Optimal Substructure: Problem can be split into subproblems

Optimal Greedy: Making change with the fewest number of coins

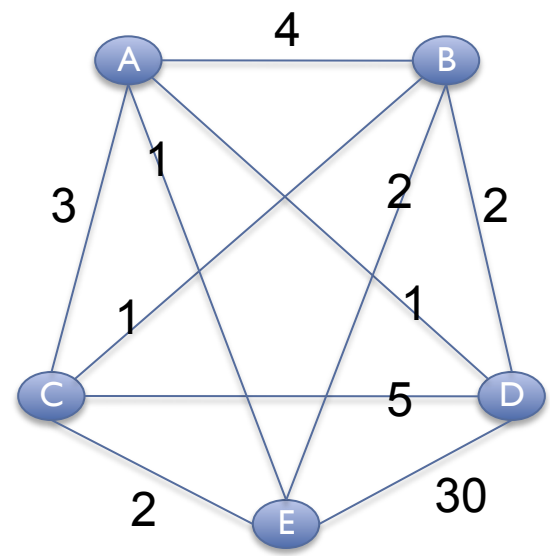
TSP Complexity

- No fast solution
  - Knowing optimal tour through n cities doesn't seem to help much for n+1 cities

[How many possible tours for n cities?]

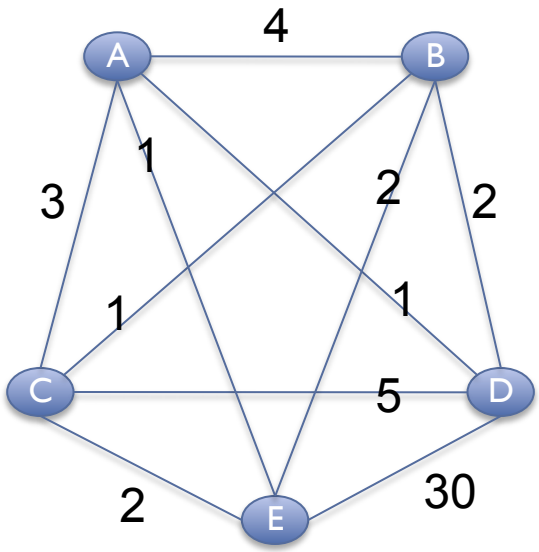


- Extensive searching is the only provably correct algorithm
  - Brute Force: O(n!)
    - ~20 cities max
    - $20! = 2.4 \times 10^{18}$



Branch-and-Bound

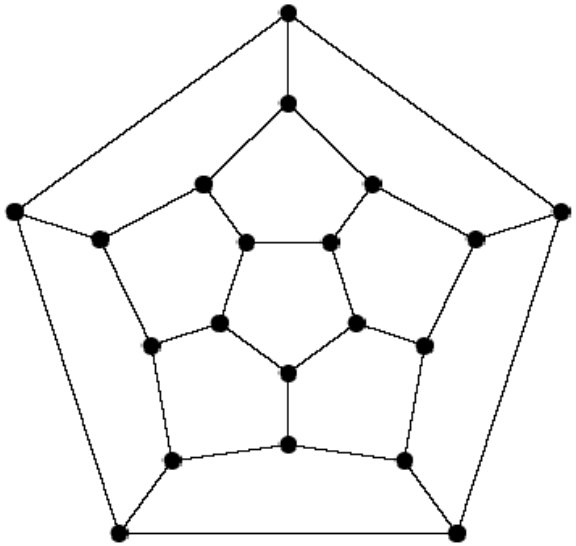
- Abort on suboptimal solutions as soon as possible
  - ADBECA = 1+2+2+2+3 = 10
  - ABDE = 4+2+30 > 10
  - ADE = 1+30 > 10
  - AED = 1+30 > 10
  - …



- Performance Heuristic
  - Always gives the optimal answer
  - Doesn't always help performance, but often does
  - Current TSP record holder:
    - 85,900 cities
    - 85900! = 10^{386526}

[When not?]

TSP and NP-complete



- TSP is one of many extremely hard problems of the class NP-complete
  - Extensive searching is the only way to find an exact solution
  - Often have to settle for approx. solution

- WARNING: Many biological problems are in this class
  - Find a tour the visits every node once (Genome Assembly)
  - Find the smallest set of vertices covering the edges (Essential Genes)
  - Find the largest clique in the graph (Protein Complexes)
  - Find the highest mutual information encoding scheme (Neurobiology)
  - Find the best set of moves in tetris
  - …
  - http://en.wikipedia.org/wiki/List_of_NP-complete_problems

Break


PHASE TWO : INTERPRETATION
I THINK I FOUND A CORNER PIECE.
3 BILLION PIECES
GENOME

What is your genome?

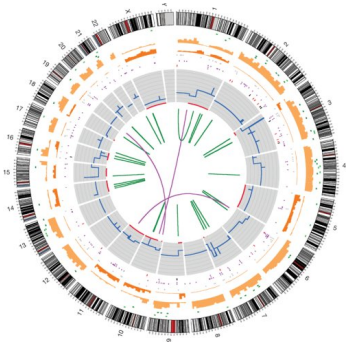
Like Dickens, we must computationally reconstruct a genome from short fragments

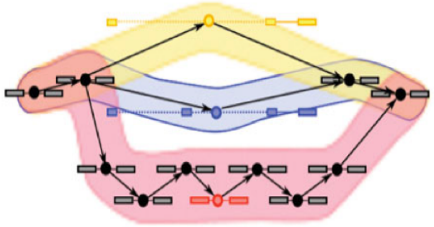
Assembly Applications

- Novel genomes

- Metagenomes

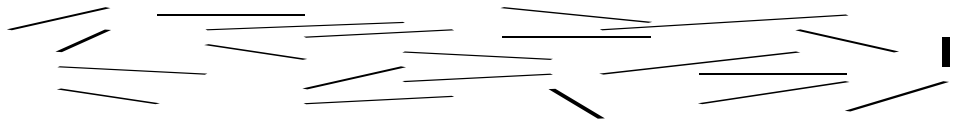
- Sequencing assays
  - Structural variations
  - Transcript assembly
  - …





Assembling a Genome

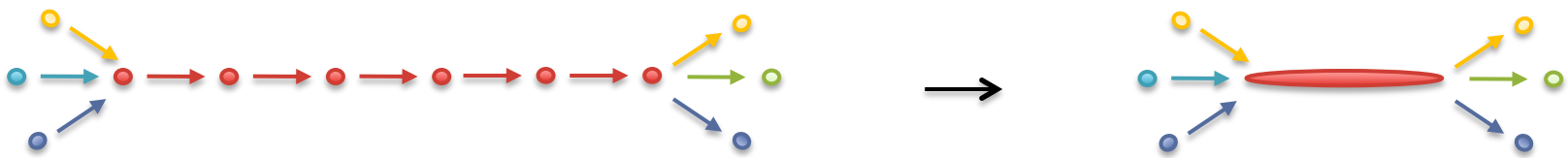
1. Shear & Sequence DNA



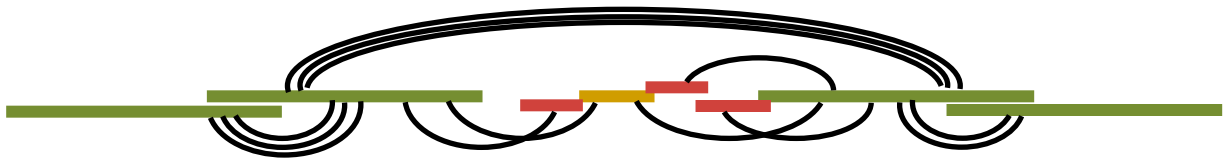
2. Construct assembly graph from overlapping reads

...AGCCTAGGGATGCGCGACACGT
GGATGCGCGACACGTCGCATATCCGGTTTGGTCAACCTCGGACGGAC
CAACCTCGGACGGACCTCAGCGAA...

3. Simplify assembly graph



4. Detangle graph with long reads, mates, and other links



Shortest Common Superstring

Given: $S = \{s_1, \ldots, s_n\}$

Problem: Find minimal length superstring of S

s_1 CACCC

s_2 CCGGGTGC

s_3 CCACC

$s_{1,}\mathbf{s_2},s_3$ = CAC**CCGGGTGC**CACC 15

$s_1,\mathbf{s_3},s_2$ = CAC**CCACC**GGGTGC14

$s_2,\mathbf{s_1},s_3$ = CCGGGTG**CACCC**ACC 15

$s_2,\mathbf{s_3},s_1$ = CCGGGTG**CCACC**C 13

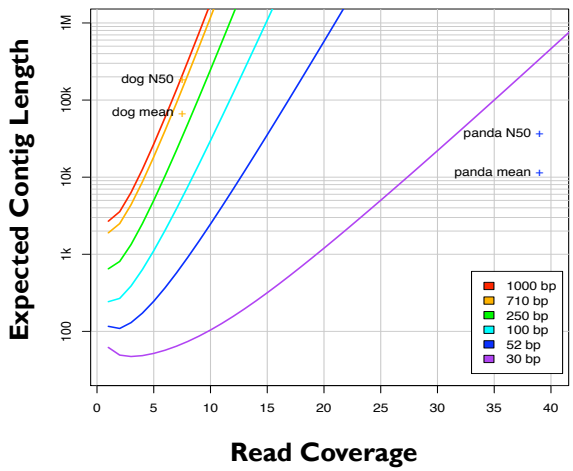
$s_3,\mathbf{s_1},s_2$ = C**CACCC**GGGTGC 12

$s_3,\mathbf{s_2},s_1$ = CCA**CCGGGTGC**ACCC 15

NP-Complete by reduction from VERTEX-COVER and later DIRECTED-HAMILTONIAN-PATH

Ingredients for a good assembly

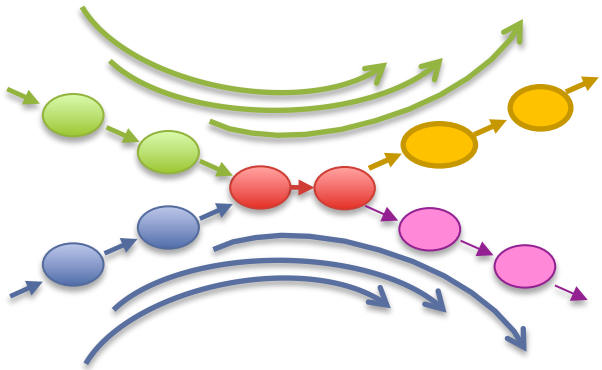
Coverage



High coverage is required

- Oversample the genome to ensure every base is sequenced with long overlaps between reads
- Biased coverage will also fragment assembly

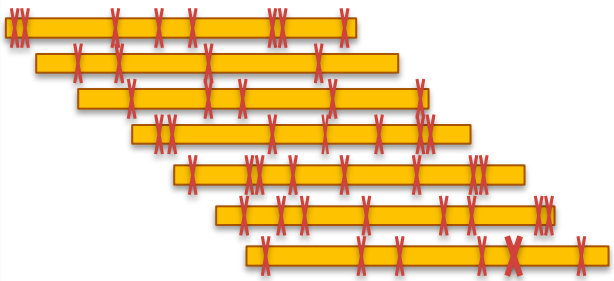
Read Length



Reads & mates must be longer than the repeats

- Short reads will have ***false overlaps*** forming hairball assembly graphs
- With long enough reads, assemble entire chromosomes into contigs

Quality

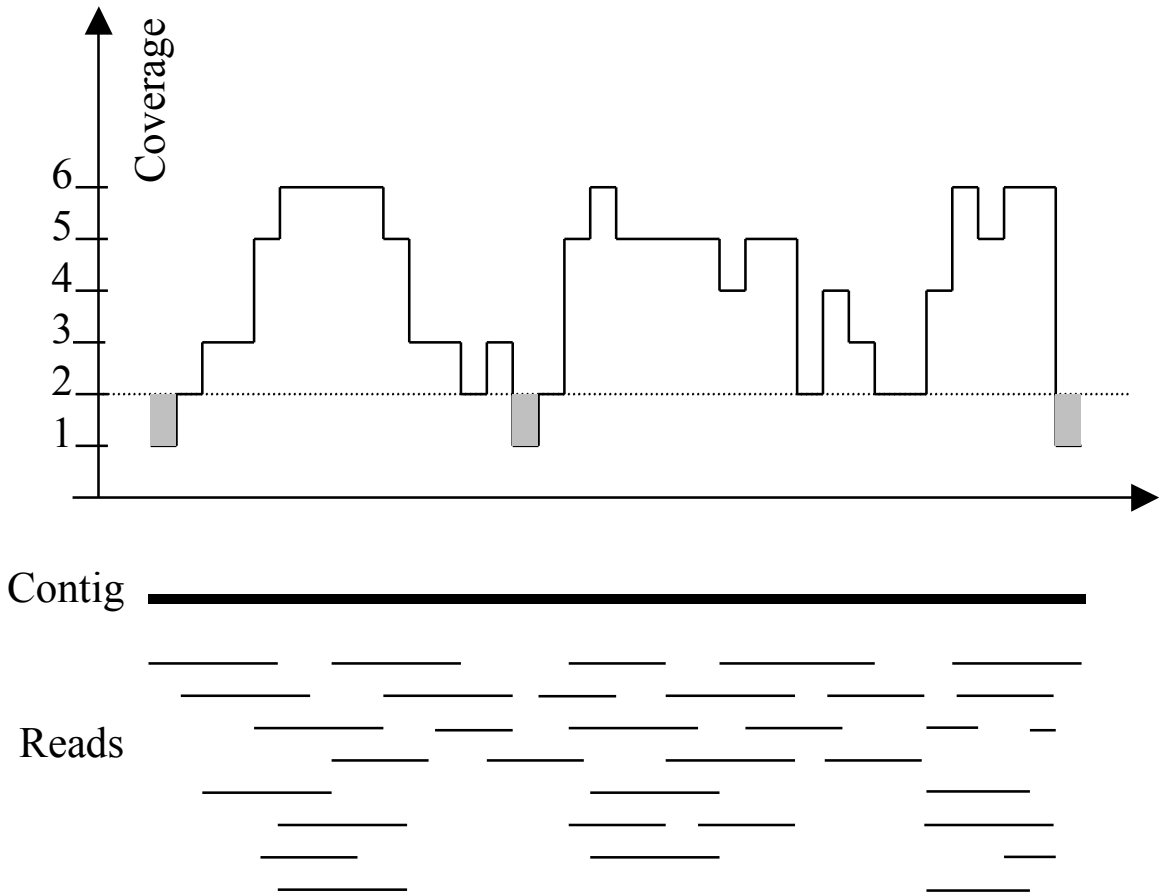


Errors obscure overlaps

- Reads are assembled by finding kmers shared in pair of reads
- High error rate requires very short seeds, increasing complexity and forming assembly hairballs

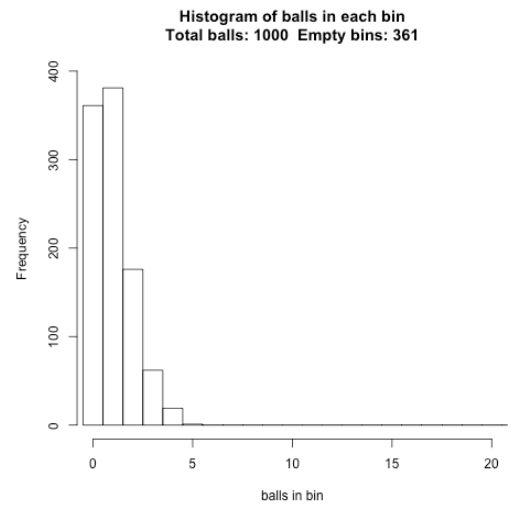
Current challenges in *de novo* plant genome sequencing and assembly
Schatz MC, Witkowski, McCombie, WR (2012) *Genome Biology*. 12:243

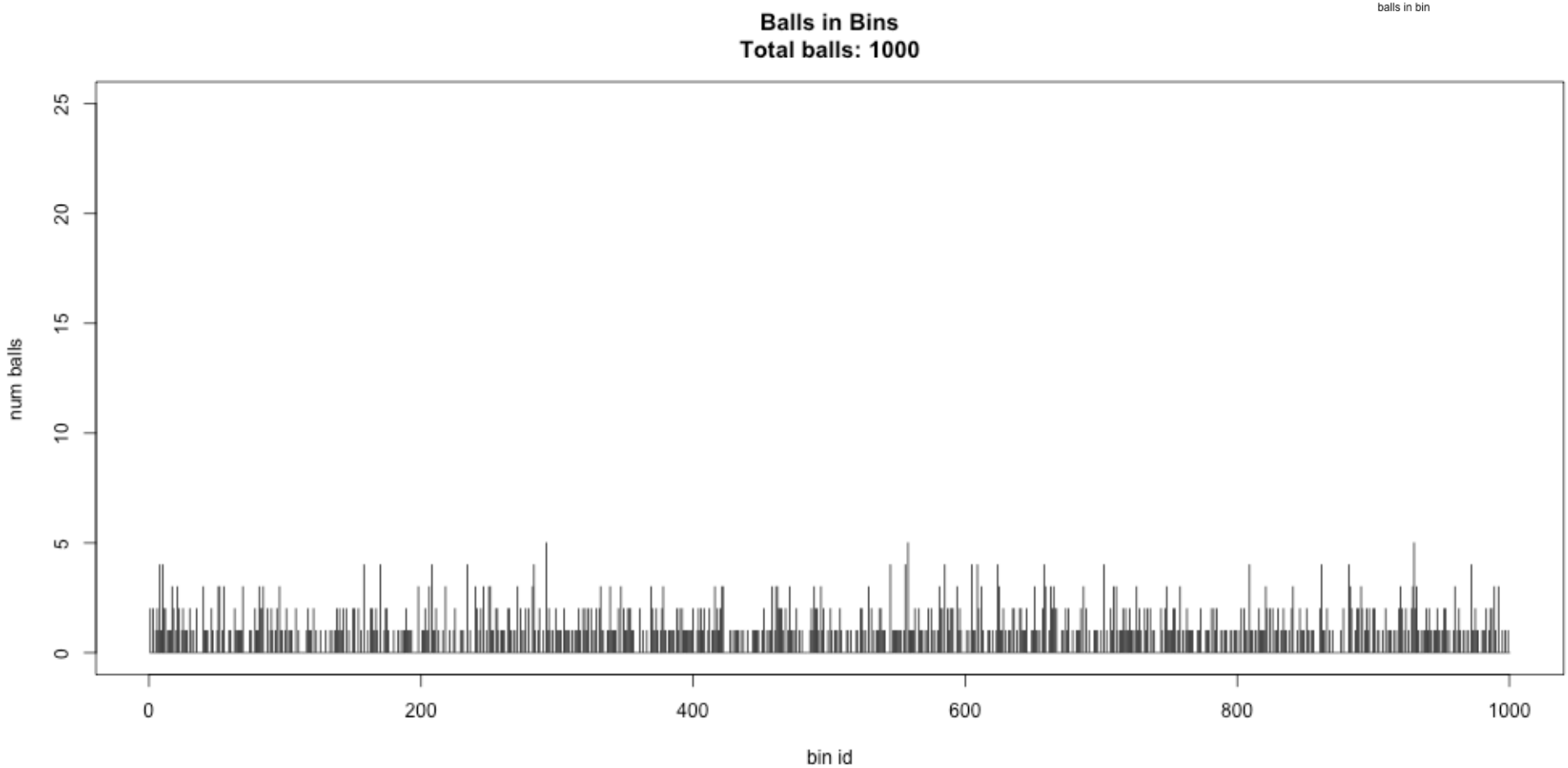
Typical contig coverage



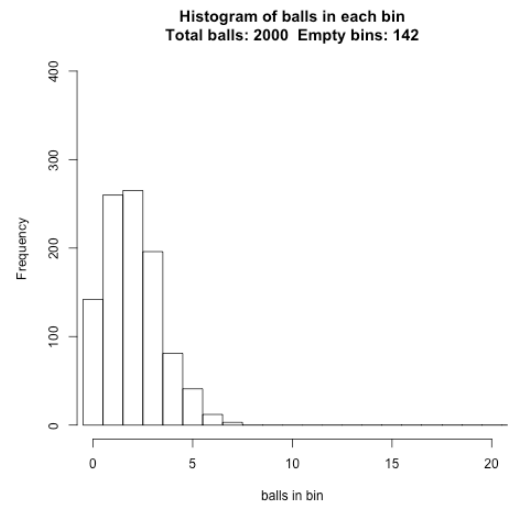
Imagine raindrops on a sidewalk

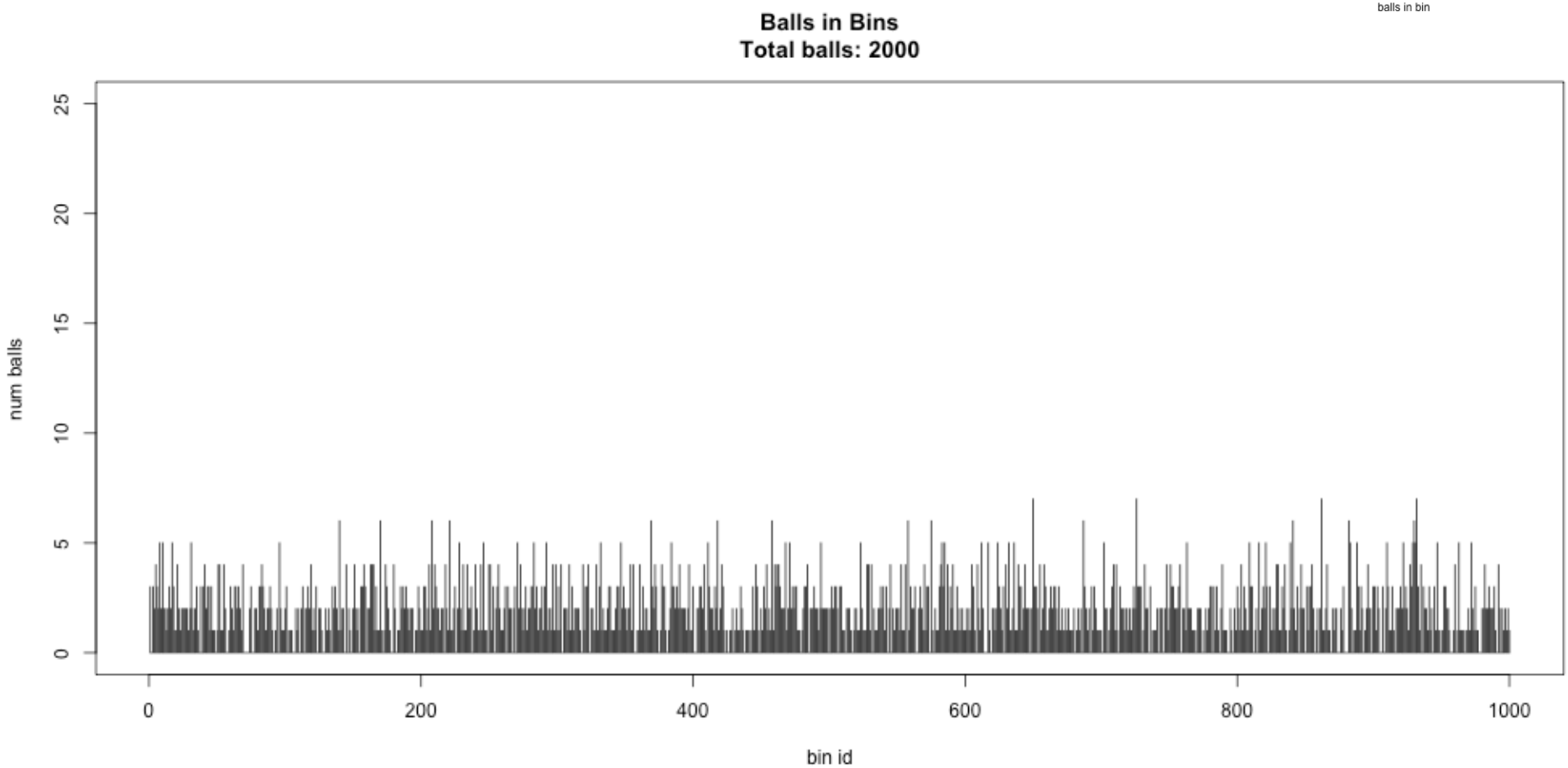
Balls in Bins 1x



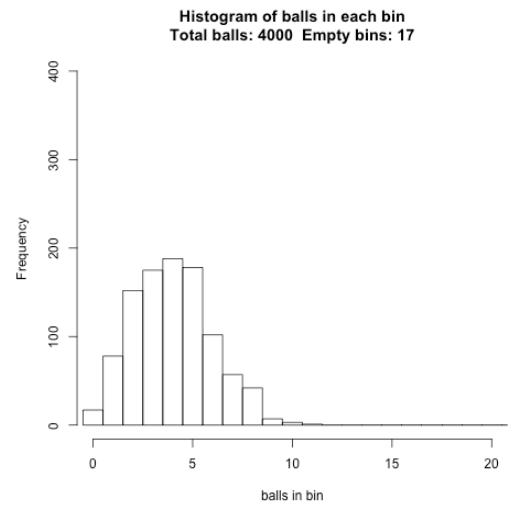


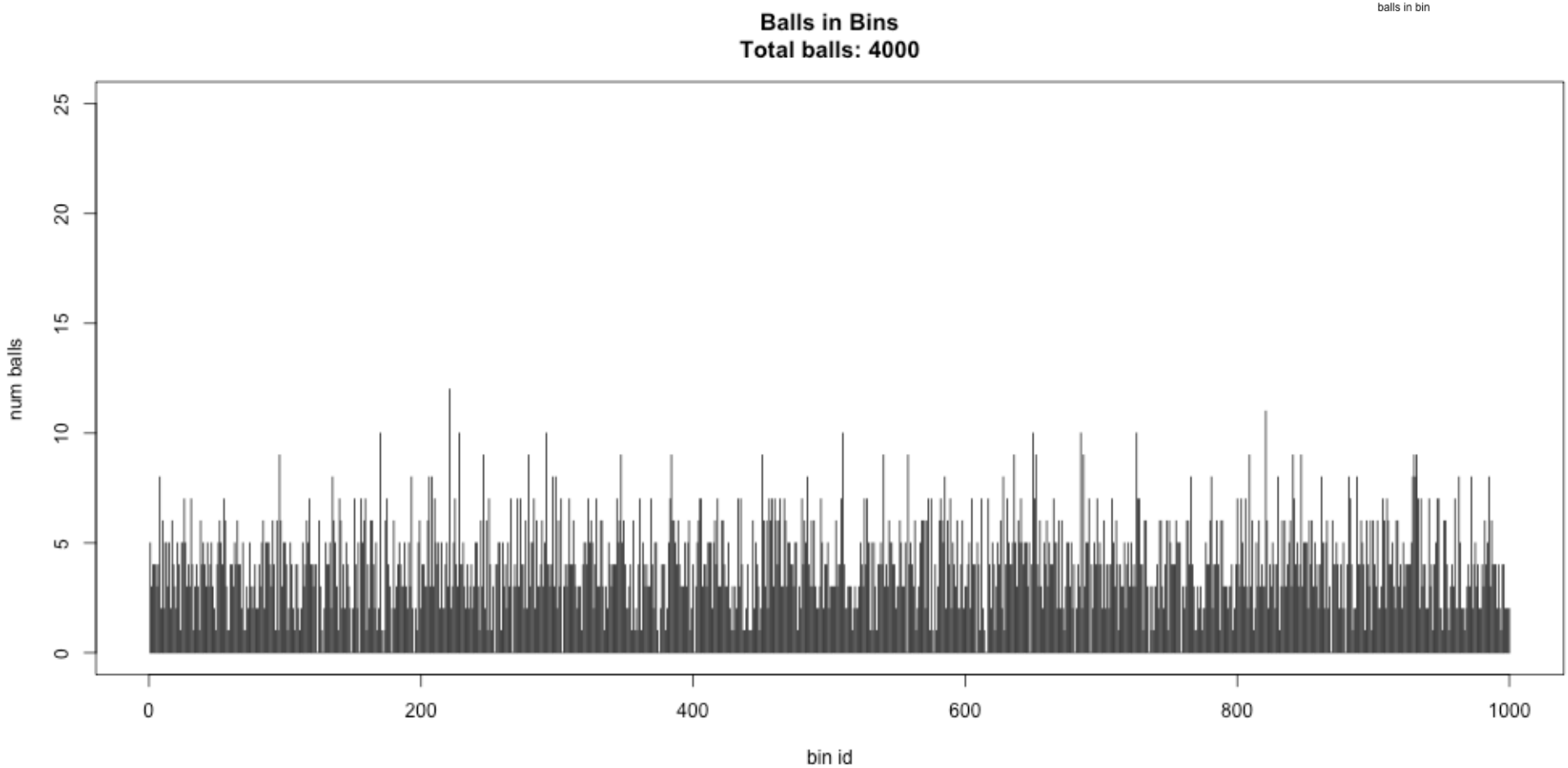
Balls in Bins 2x



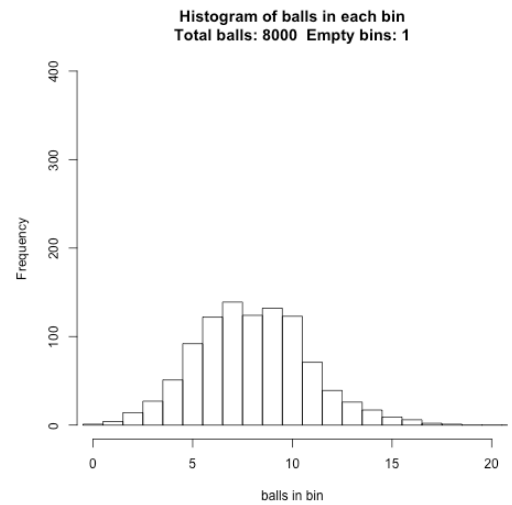


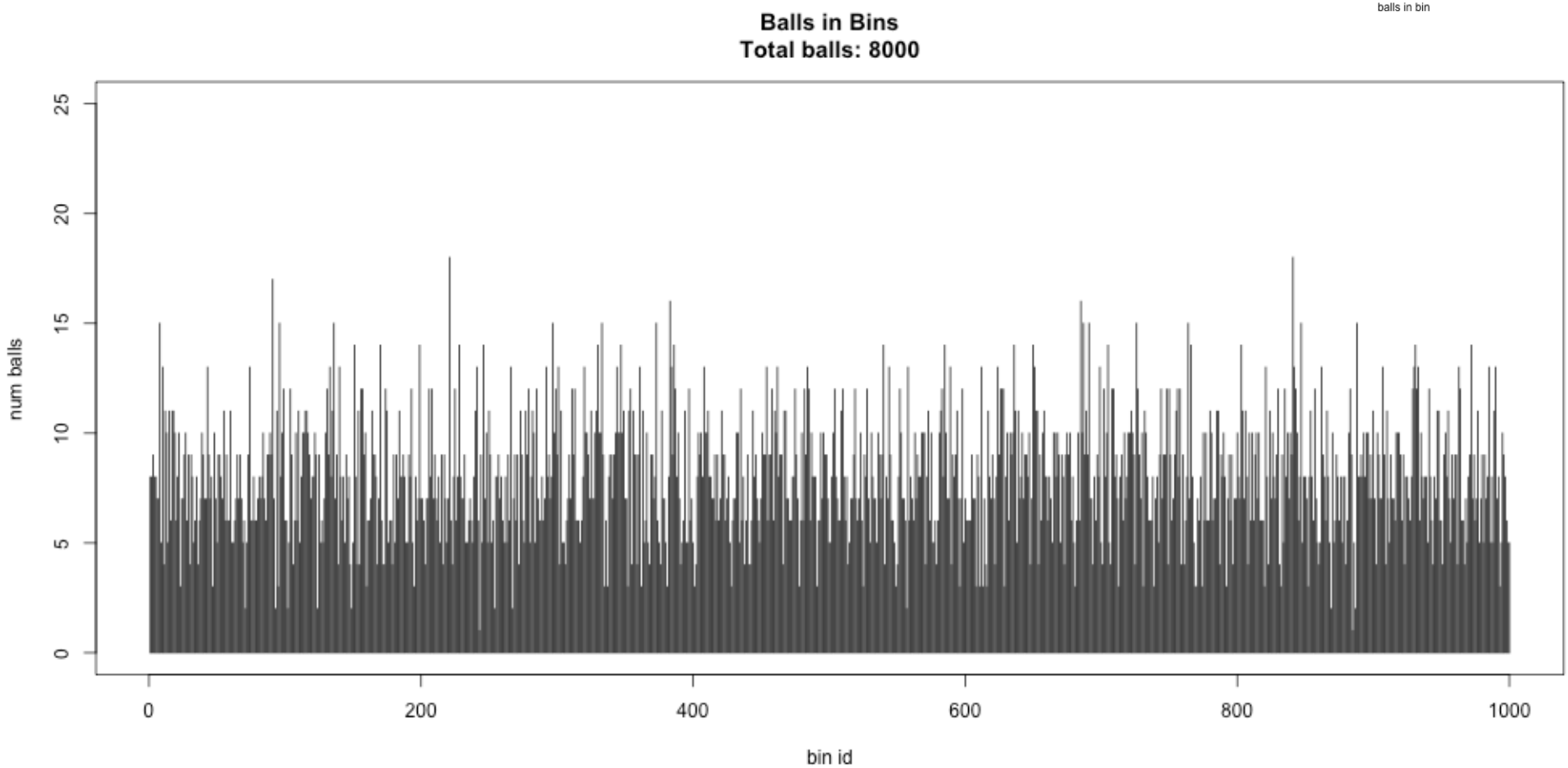
Balls in Bins 4x





Balls in Bins 8x





Lander Waterman Statistics

L = read length

T = minimum overlap

G = genome size

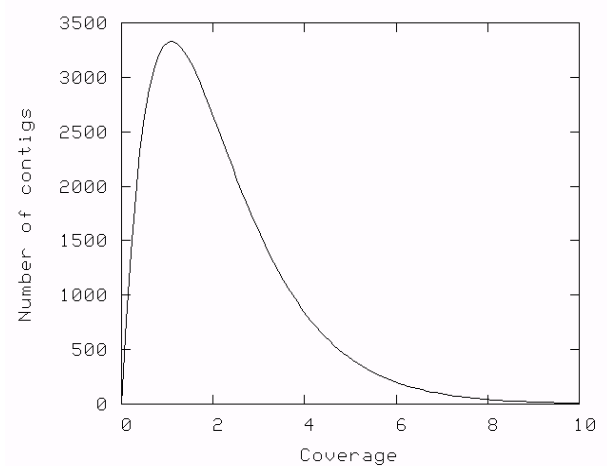
N = number of reads

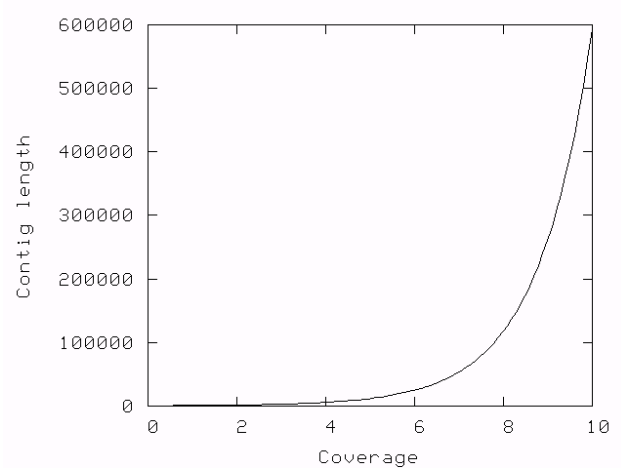
c = coverage (NL / G)

$\sigma = 1 - T/L$

$E(\#contigs) = Ne^{-c\sigma}$

$E(contig\ size) = L(e^{c\sigma} - 1) / c + 1 - \sigma$



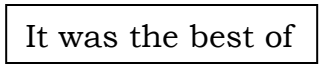


Genomic mapping by fingerprinting random clones: a mathematical analysis
Lander ES, Waterman MS (1988) *Genomics*. 2(3):231-239

de Bruijn Graph Construction

- $D_k = (V,E)$
  - V = All length-k subfragments (k < l)
  - E = Directed edges between consecutive subfragments
    - Nodes overlap by k-1 words

Original Fragment

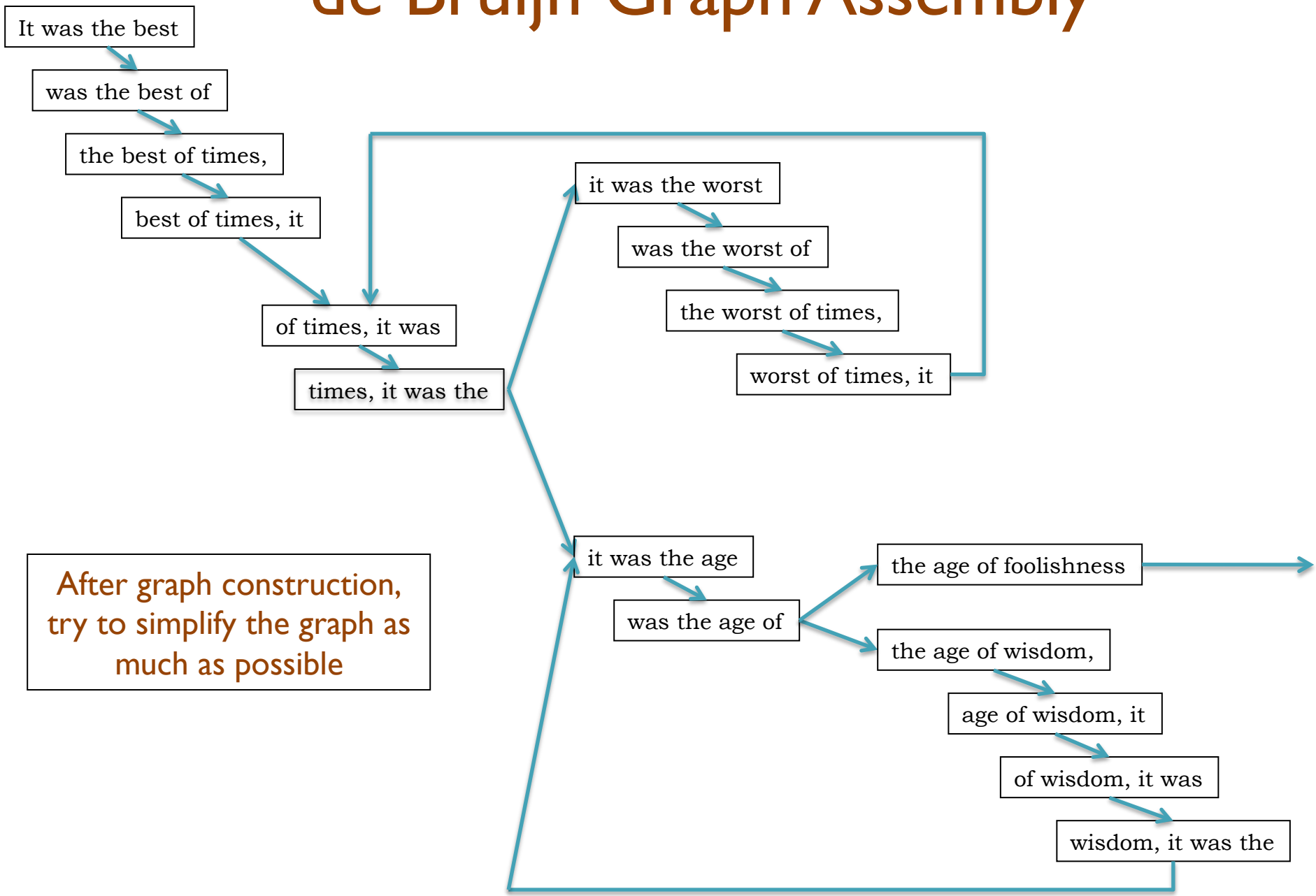


Directed Edge

- Locally constructed graph reveals the global sequence structure
  - Overlaps between sequences implicitly computed

de Bruijn, 1946
Idury and Waterman, 1995
Pevzner, Tang, Waterman, 2001

de Bruijn Graph Assembly



After graph construction, try to simplify the graph as much as possible

Overlap all pairs of sequences

overlap (19 bases)

```
...AGCCTAGACCTACAGGATGCGCGGACACGTAGC
                 GGATGCGC-GACACGTAGCTTATCCGGT...
```

% identity = 18/19 % = 94.7%

overlap - region of similarity between reads

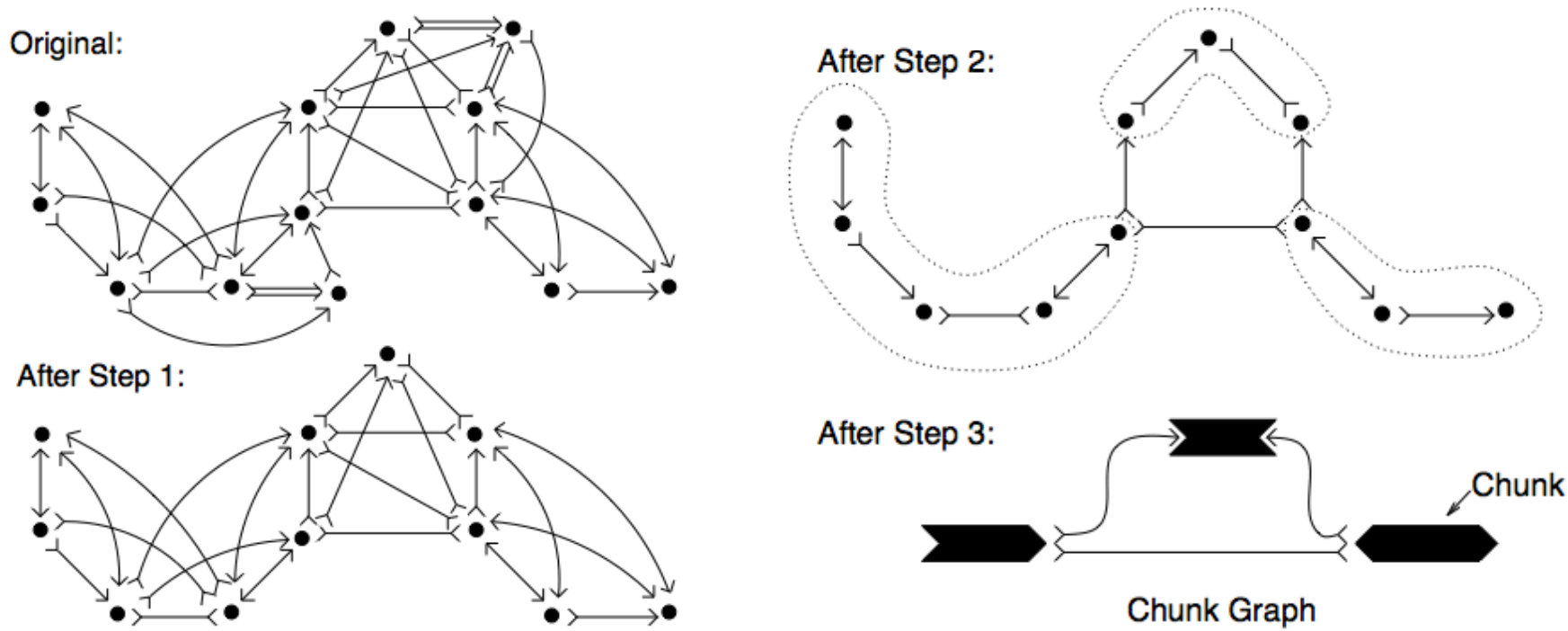
The assembler screens merges based on:

- length of overlap
- % identity in overlap region
- maximum overhang size

In practice, don't attempt to overlap all pairs, but require the pair to share an exact seed

[How do we score the overlap?]

Reducing overlaps to unitigs



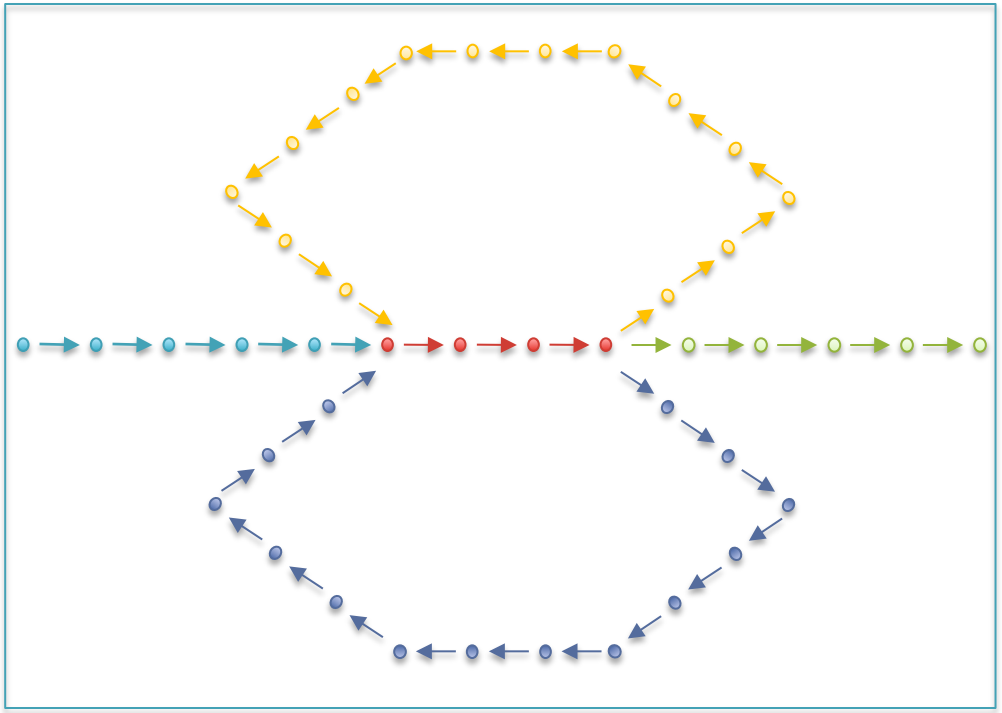
- Because we oversample the genome, most overlaps are redundant.
- Remove all "transitively inferred" overlaps
  - If A overlaps B, and B overlaps C, the extra overlap between A and C can be transitively implied

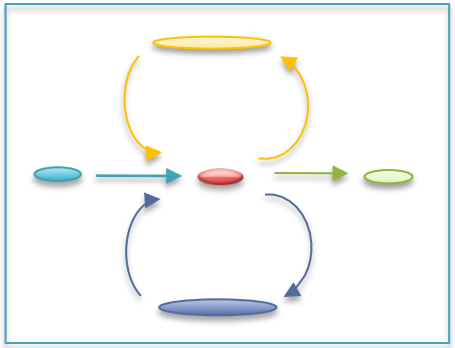
Toward simplifying and accurately formulating fragment assembly.
Myers, EW (1995) J Comp Bio. 2(2):275-90.

Unitigging / Unipathing

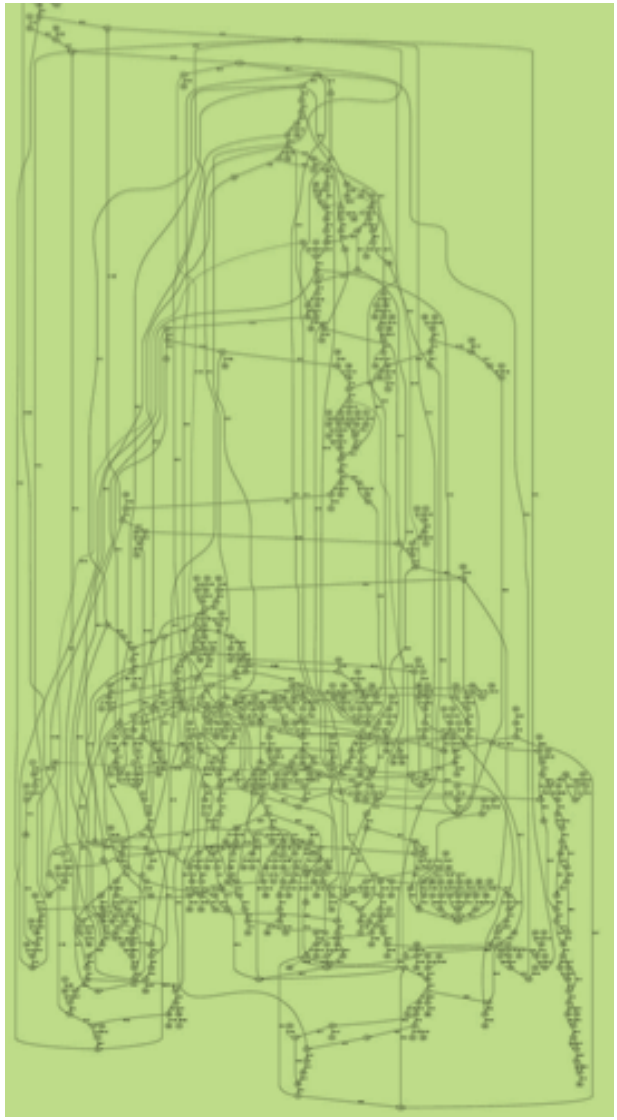
After simplification and correction, compress graph down to its non-branching initial contigs

- Aka "unitigs", "unipaths"
- Unitigs end because of (1) lack of coverage, (2) errors, and (3) repeats

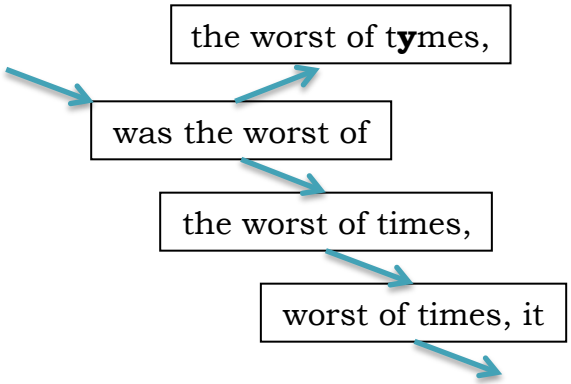
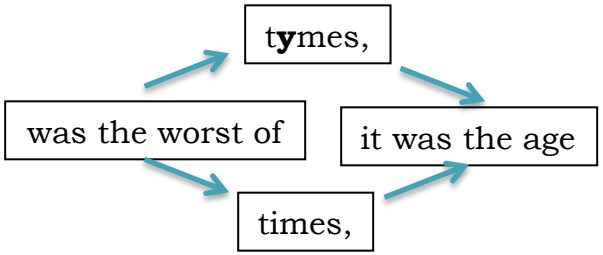




Errors in the graph



(Chaisson, 2009)

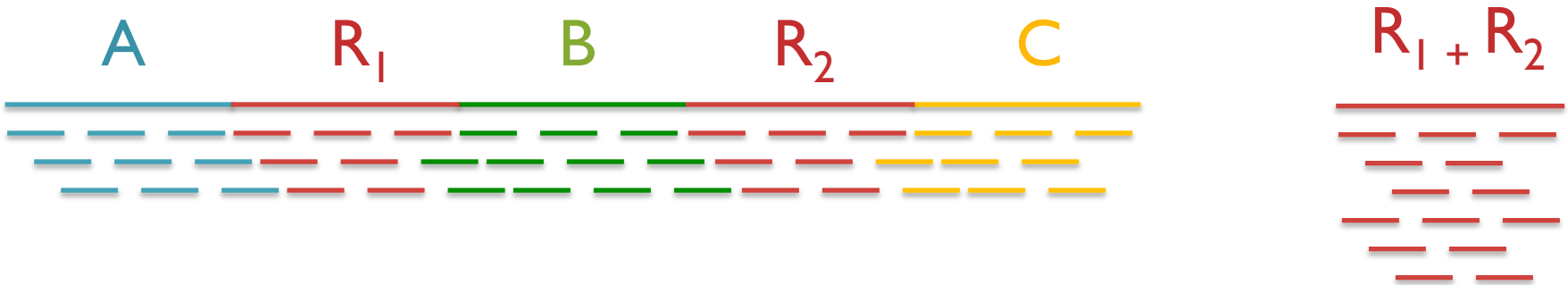
Clip Tips	Pop Bubbles
was the worst of times, was the worst of t**y**mes, the worst of times, it	was the worst of times, was the worst of t**y**mes, times, it was the age t**y**mes, it was the age
	

Repetitive regions

Repeat Type	Definition / Example	Prevalence
Low-complexity DNA / Microsatellites	$(b_1 b_2 \ldots b_k)^N$ where $1 \leq k \leq 6$ CACACACACACACACACACA	2%
SINEs (Short Interspersed Nuclear Elements)	*Alu* sequence (~280 bp) Mariner elements (~80 bp)	13%
LINEs (Long Interspersed Nuclear Elements)	~500 – 5,000 bp	21%
LTR (long terminal repeat) retrotransposons	Ty1-copia, Ty3-gypsy, Pao-BEL (~100 – 5,000 bp)	8%
Other DNA transposons		3%
Gene families & segmental duplications		4%

- Over 50% of mammalian genomes are repetitive
  - Large plant genomes tend to be even worse
  - Wheat: 16 Gbp; Pine: 24 Gbp

Repeats and Coverage Statistics



- If *n* reads are a uniform random sample of the genome of length *G*, we expect $k=n\Delta/G$ reads to start in a region of length Δ.
  - If we see many more reads than k (if the arrival rate is > A) , it is likely to be a collapsed repeat

$$\Pr(X-copy)=\binom{n}{k}\left(\frac{X\Delta}{G}\right)^{k}\left(\frac{G-X\Delta}{G}\right)^{n-k}$$

$$A(\Delta,k)=\ln\left(\frac{\Pr(1-copy)}{\Pr(2-copy)}\right)=\ln\left(\frac{\frac{(\Delta n/G)^{k}}{k!}e^{\frac{-\Delta n}{G}}}{\frac{(2\Delta n/G)^{k}}{k!}e^{\frac{-2\Delta n}{G}}}\right)=\frac{n\Delta}{G}-k\ln 2$$

The fragment assembly string graph
Myers, EW (2005) Bioinformatics. 21(suppl 2): ii79-85.

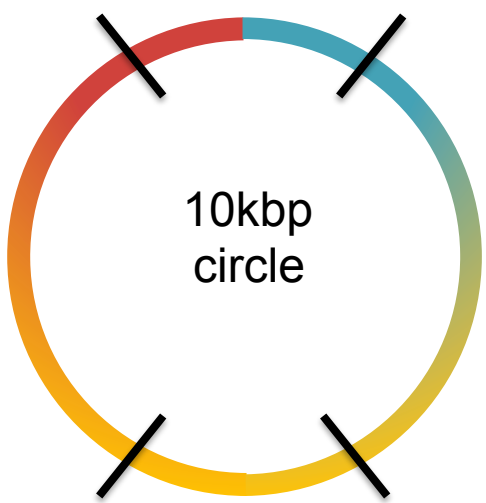
Paired-end and Mate-pairs

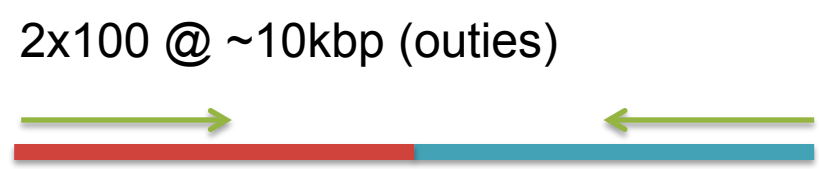
Paired-end sequencing

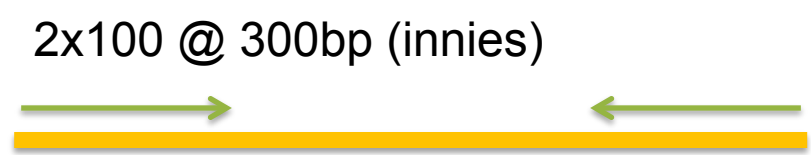
- Read one end of the molecule, flip, and read the other end
- Generate pair of reads separated by up to 500bp with inward orientation

Mate-pair sequencing

- Circularize long molecules (1-10kbp), shear into fragments, & sequence
- Mate failures create short paired-end reads

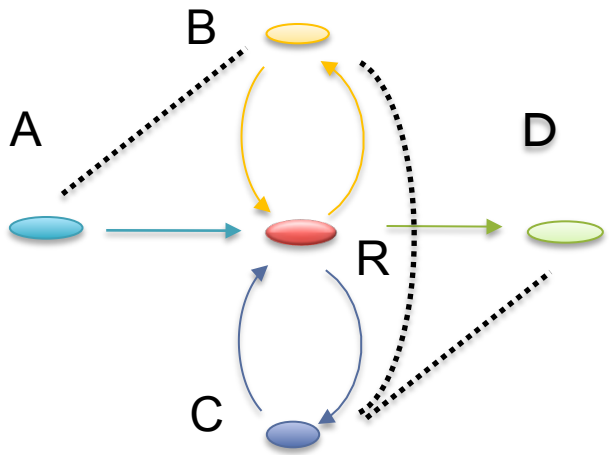




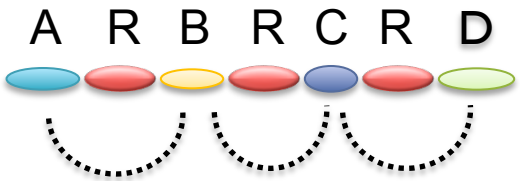


Scaffolding

- Initial contigs (*aka* unipaths, unitigs) terminate at
  - *Coverage gaps*: especially extreme GC
  - *Conflicts*: errors, repeat boundaries



- Use mate-pairs to resolve correct order through assembly graph
  - Place sequence to satisfy the mate constraints
  - Mates through repeat nodes are tangled

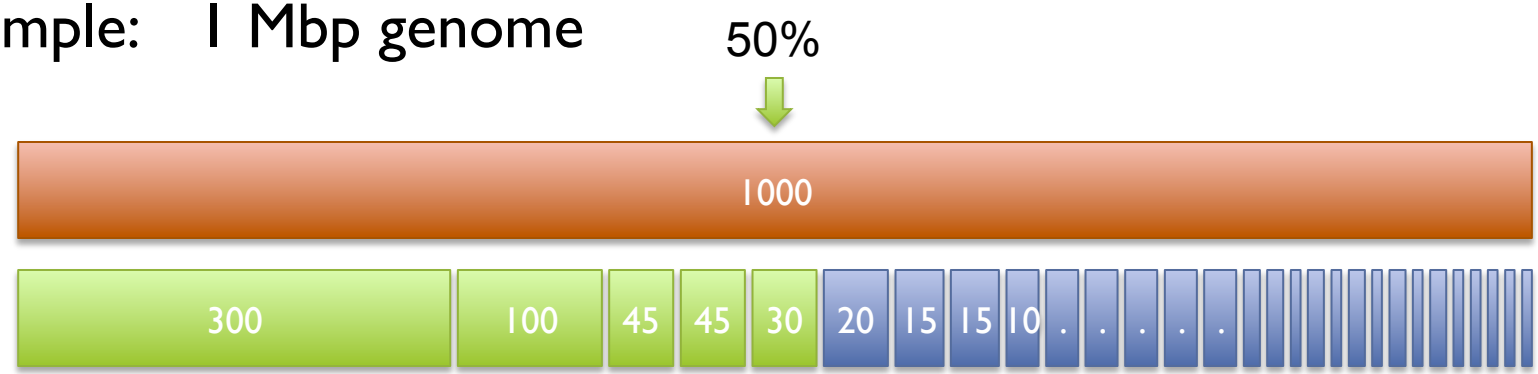


- Final scaffold may have internal gaps called sequencing gaps
  - We know the order, orientation, and spacing, but just not the bases. Fill with Ns instead

N50 size

Def: 50% of the genome is in contigs as large as the N50 value

Example: 1 Mbp genome



N50 size = 30 kbp

(300k+100k+45k+45k+30k = 520k >= 500kbp)

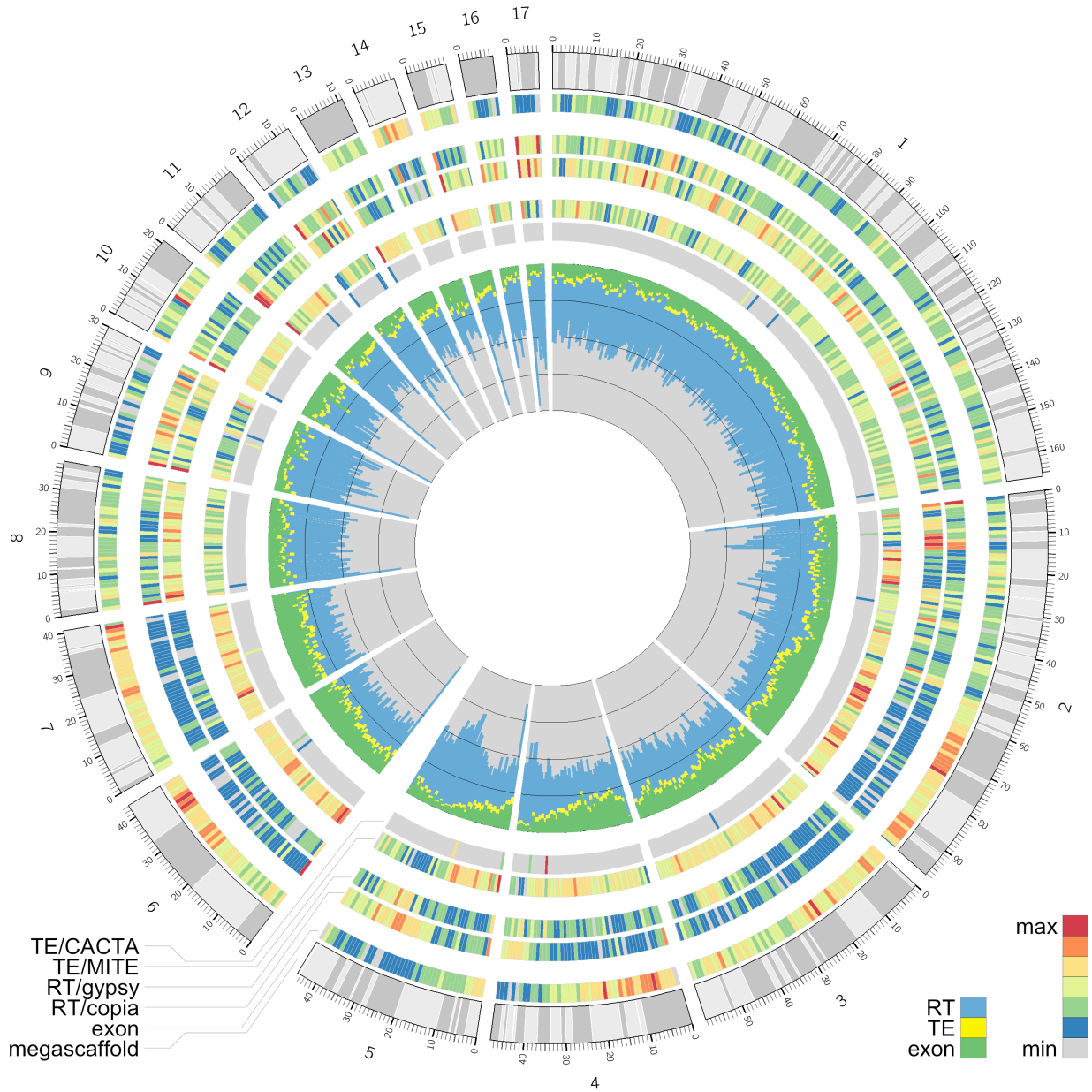
Note:

N50 values are only meaningful to compare when base genome size is the same in all cases

Publishing a genome!

After assembly:

- Validation
- WGA
- BLAST
- CEGMA
- Gene Finding
- Repeat mask
- RNA-seq
- *-seq
- …
- Publish! ☺

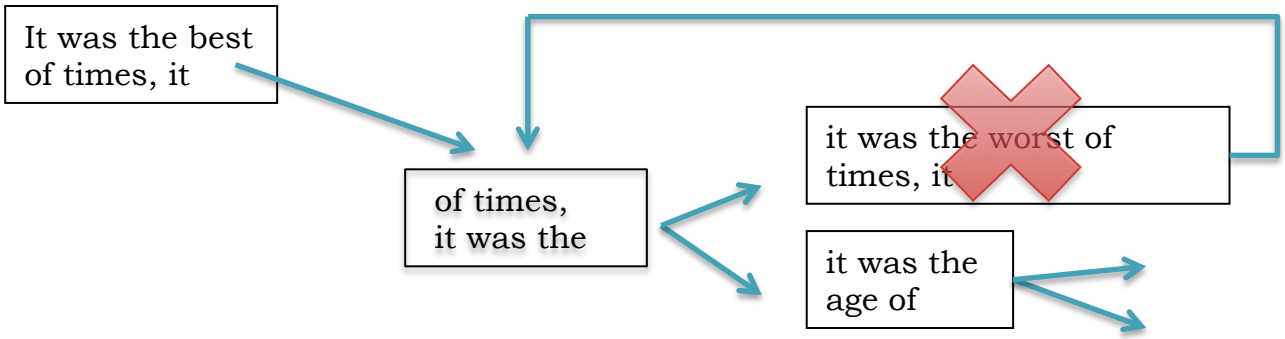


Assembly Validation

Automatically scan an assembly to locate misassembly signatures for further analysis and correction

Assembly-validation pipeline

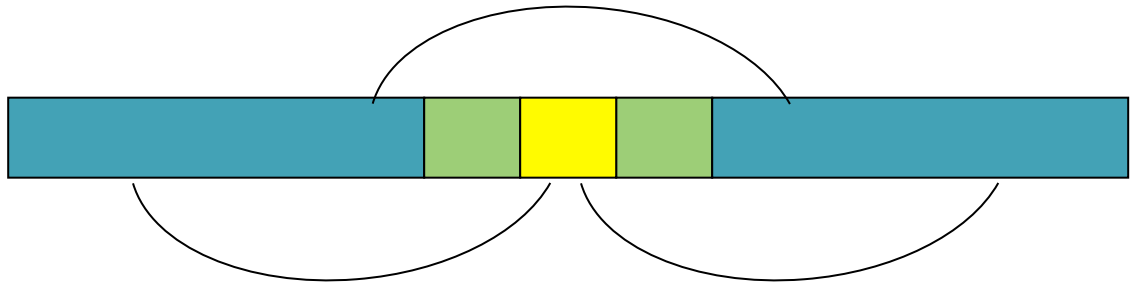
1. Evaluate Mate Pairs & Libraries
2. Evaluate Read Alignments
3. Evaluate Read Breakpoints
4. Analyze Depth of Coverage



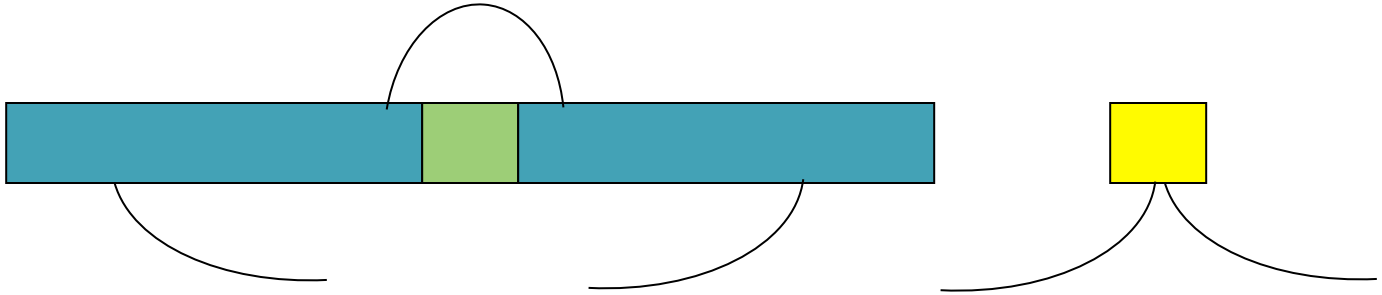
Genome Assembly forensics: finding the elusive mis-assembly.
Phillippy, AM, Schatz, MC, Pop, M. (2008) *Genome Biology* 9:R55.

Mate-Happiness: asmQC

- Excision: Skip reads between flanking repeats
  - Truth



  - Misassembly: Compressed Mates, Missing Mates



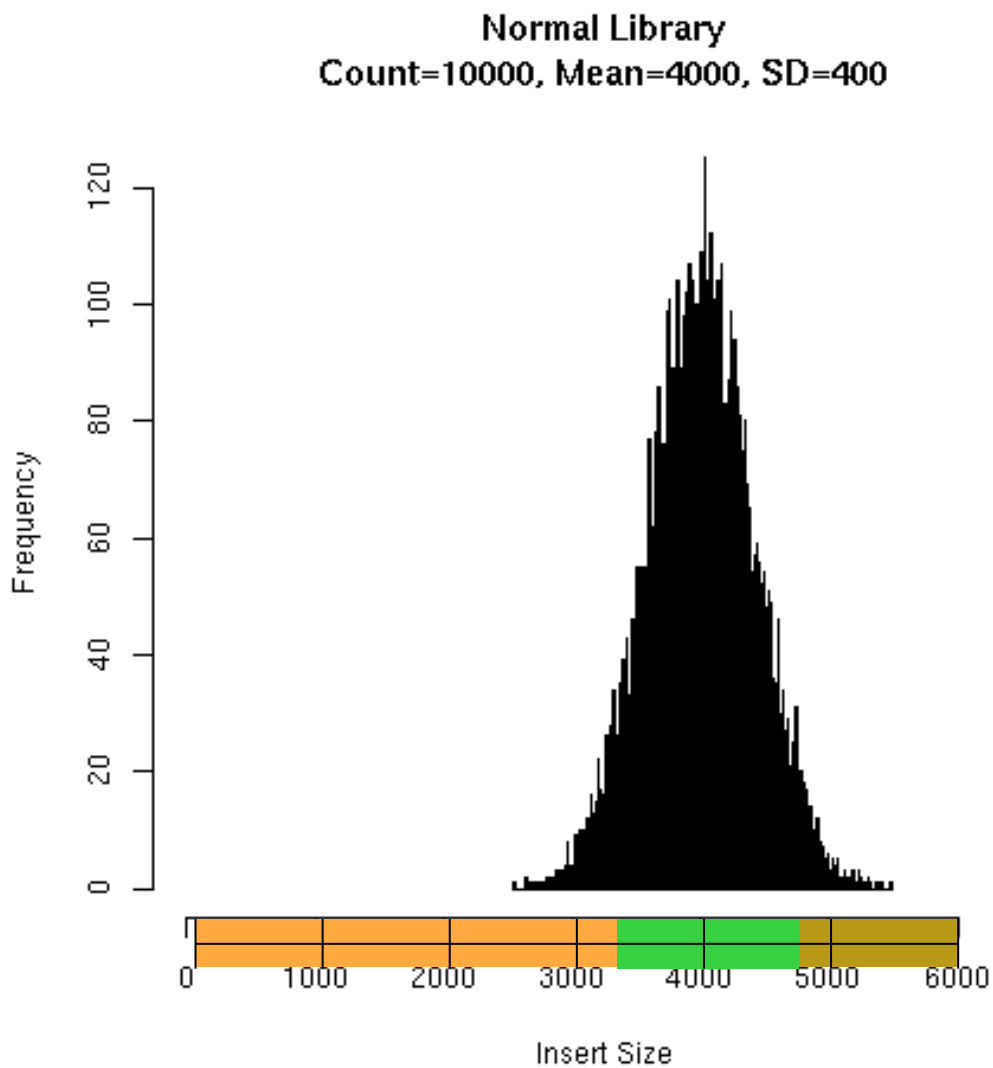
Forensics

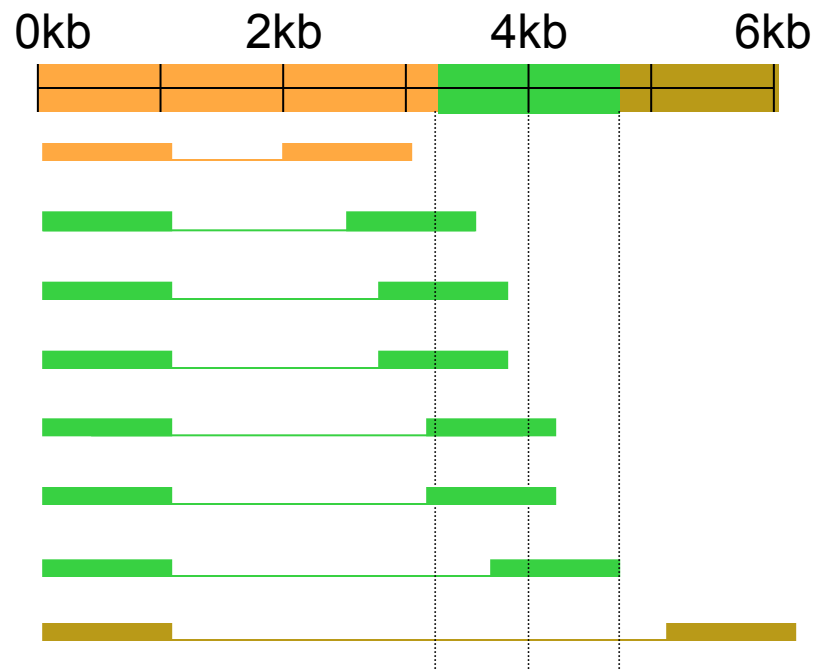
C/E Statistic

- The presence of individual compressed or expanded mates is rare but expected.
- Do the inserts spanning a given position differ from the rest of the library?
  - Flag large differences as potential misassemblies
  - Even if each individual mate is "happy"
- Compute the statistic at all positions
  - (Local Mean – Global Mean) / Scaling Factor
- Introduced by Jim Yorke's group at UMD

Forensics

Sampling the Genome





8 inserts: 3kb-6kb

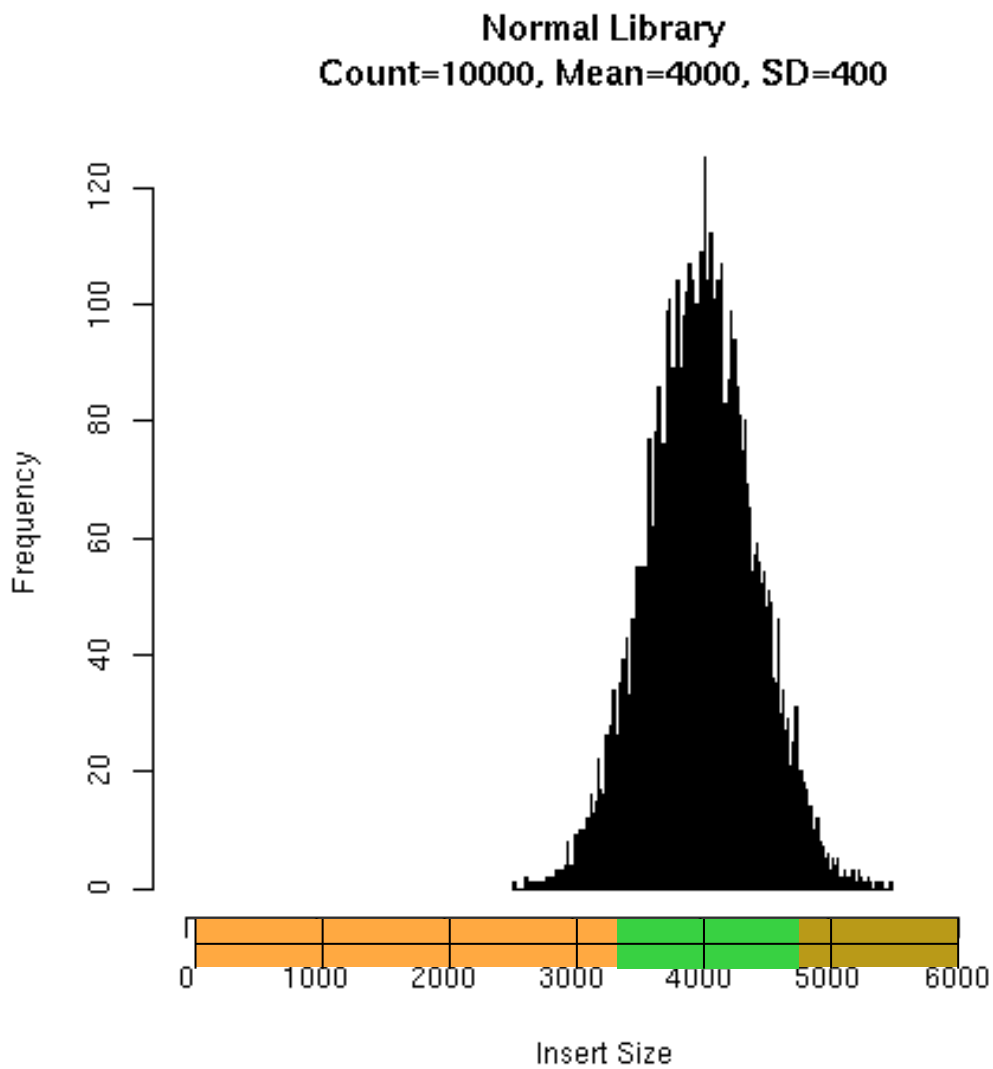
Local Mean: 4048

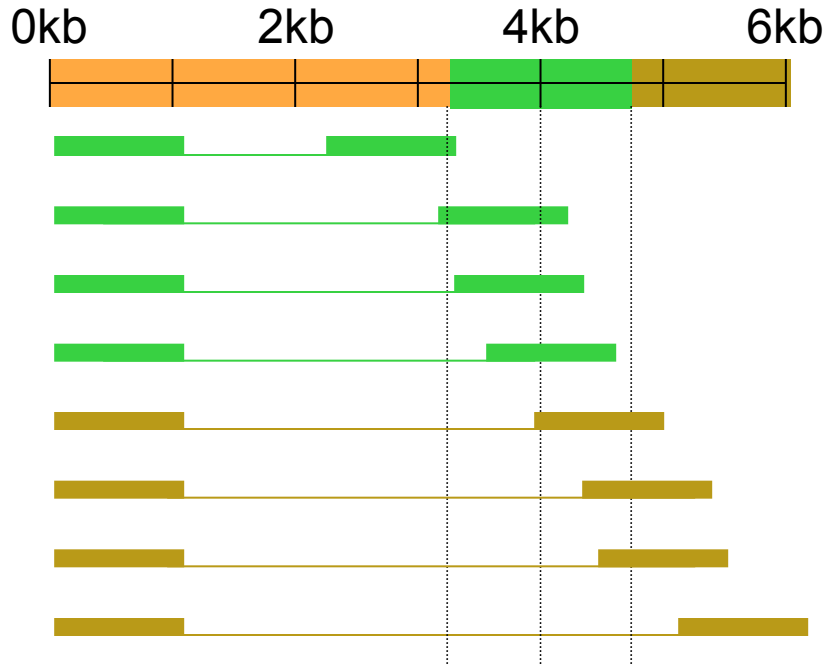
C/E Stat: $\frac{(4048-4000)}{(400 / \sqrt{8})}$ = +0.33

Near 0 indicates overall happiness

Forensics

C/E-Statistic: Expansion





8 inserts: 3.2kb-6kb

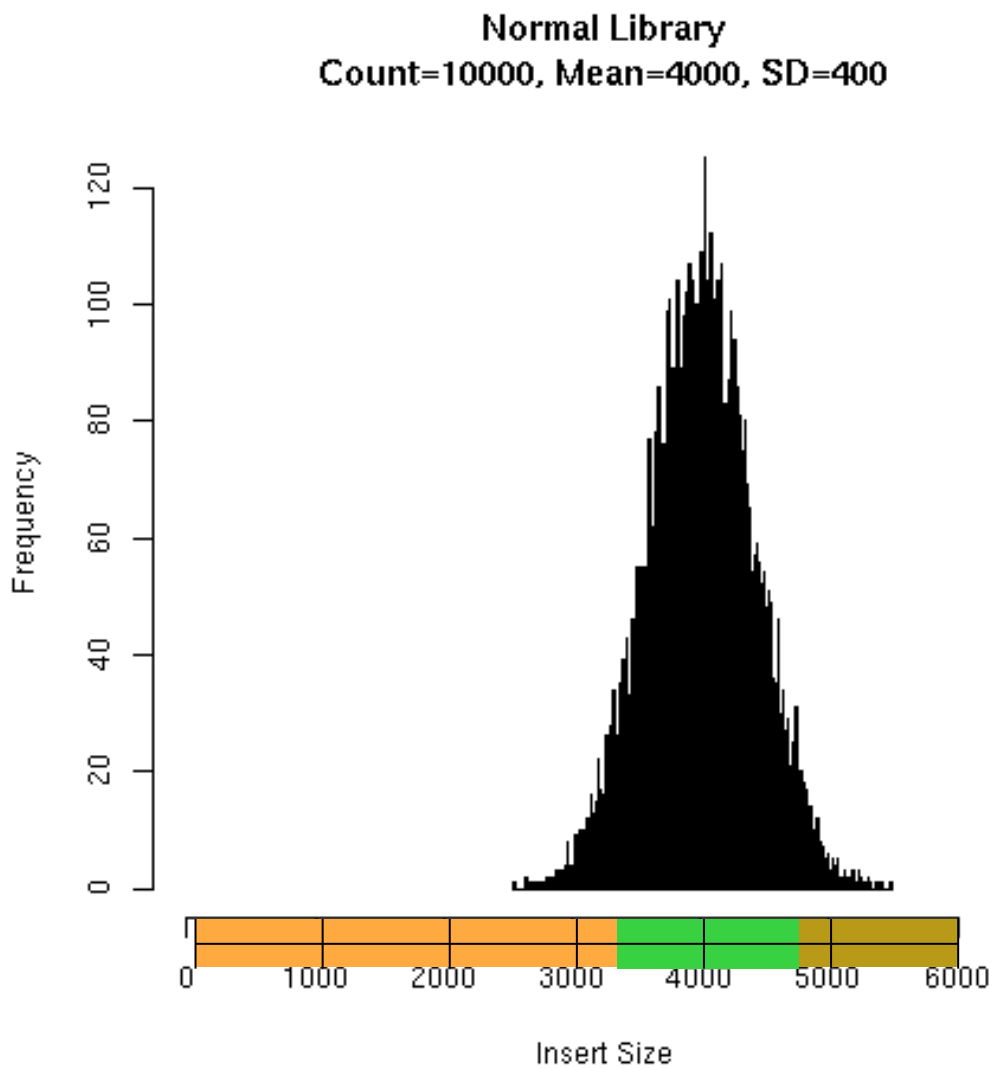
Local Mean: 4461

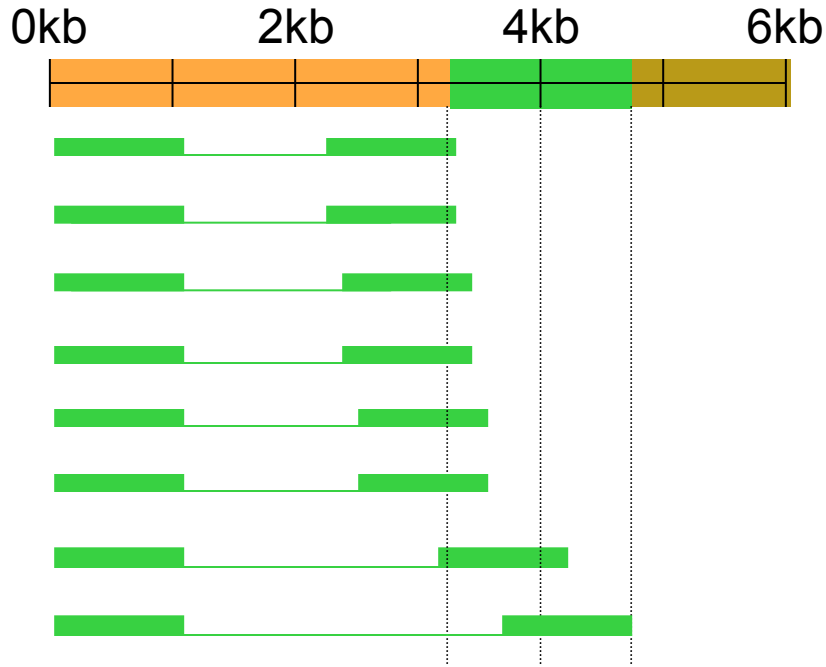
C/E Stat: $\frac{(4461-4000)}{(400 / \sqrt{8})}$ = +3.26

C/E Stat ≥ 3.0 indicates Expansion

Forensics

C/E-Statistic: Compression





8 inserts: 3.2 kb-4.8kb

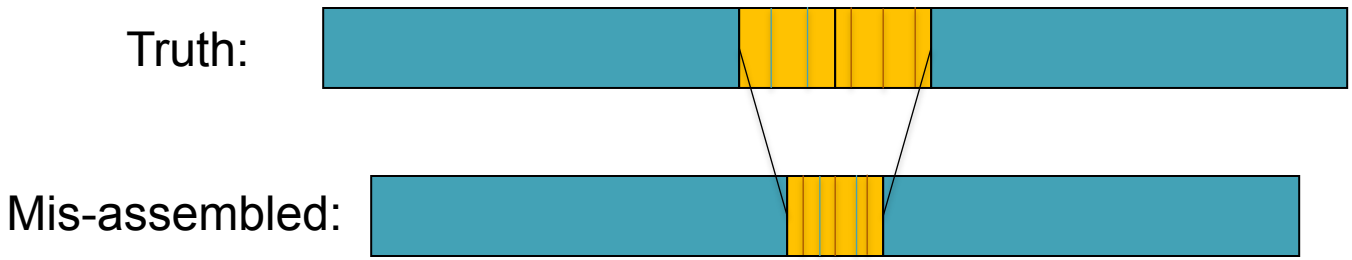
Local Mean: 3488

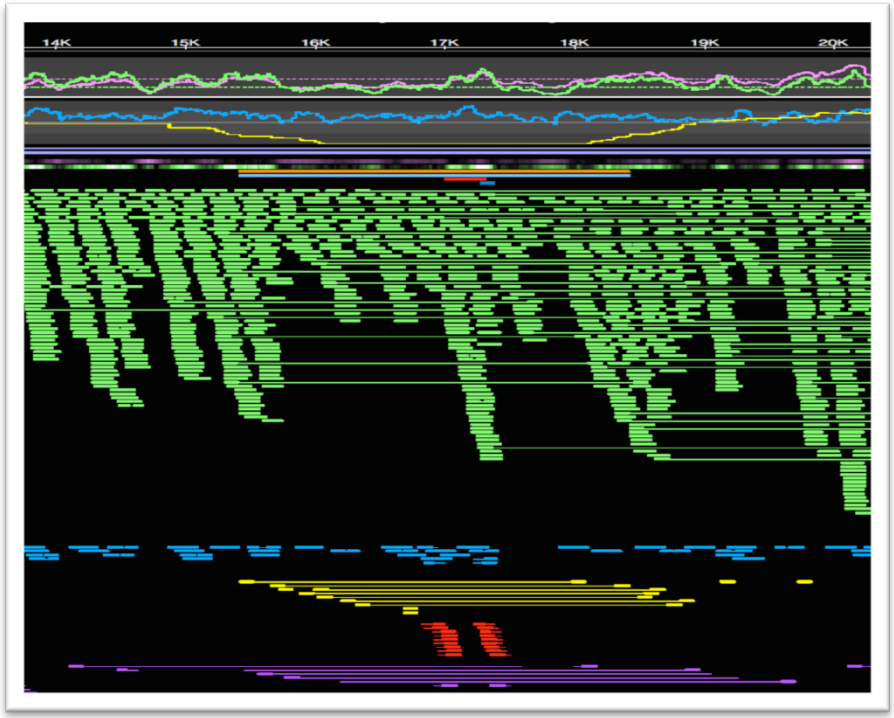
C/E Stat: $\frac{(3488-4000)}{(400 / \sqrt{8})}$ = -3.62

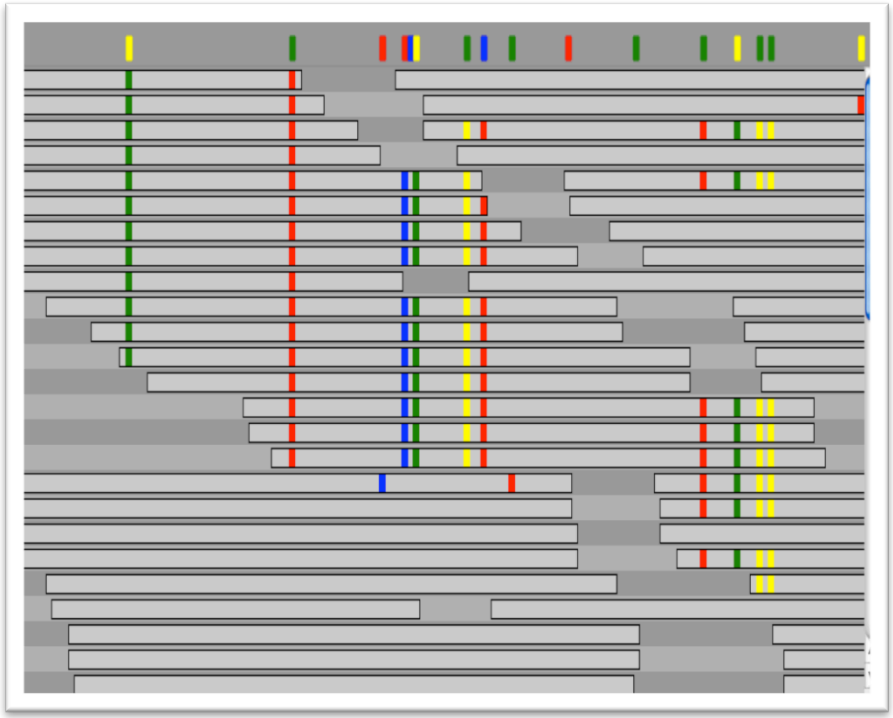
C/E Stat ≤ -3.0 indicates Compression

Forensics

Assembly Forensics







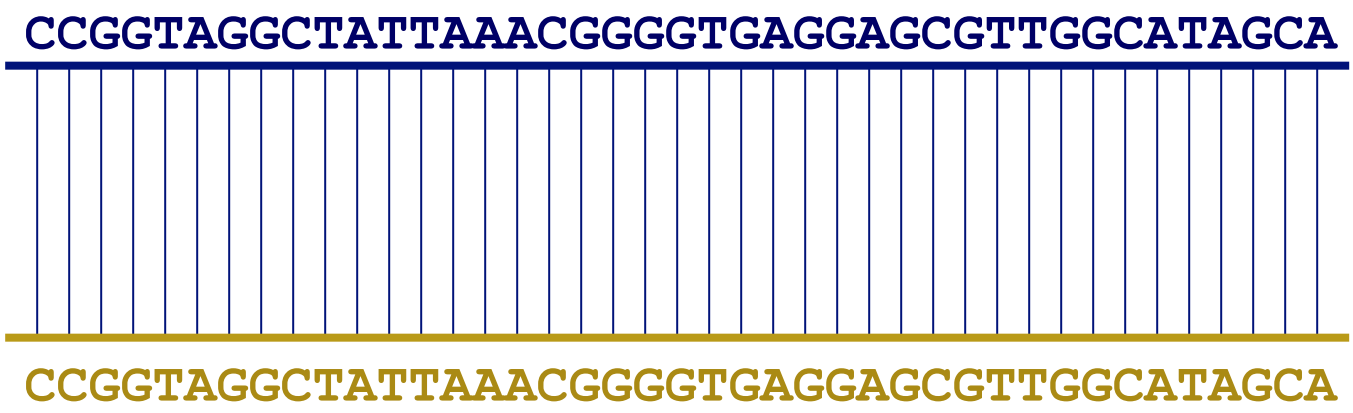
Hawkeye & AMOS: Visualizing and assessing the quality of genome assemblies
Schatz, M.C. *et al.* (2011) *Briefings in Bioinformatics.*

Whole Genome Alignment with MUMmer

Slides Courtesy of Adam M. Phillippy
amp@umics.umd.edu

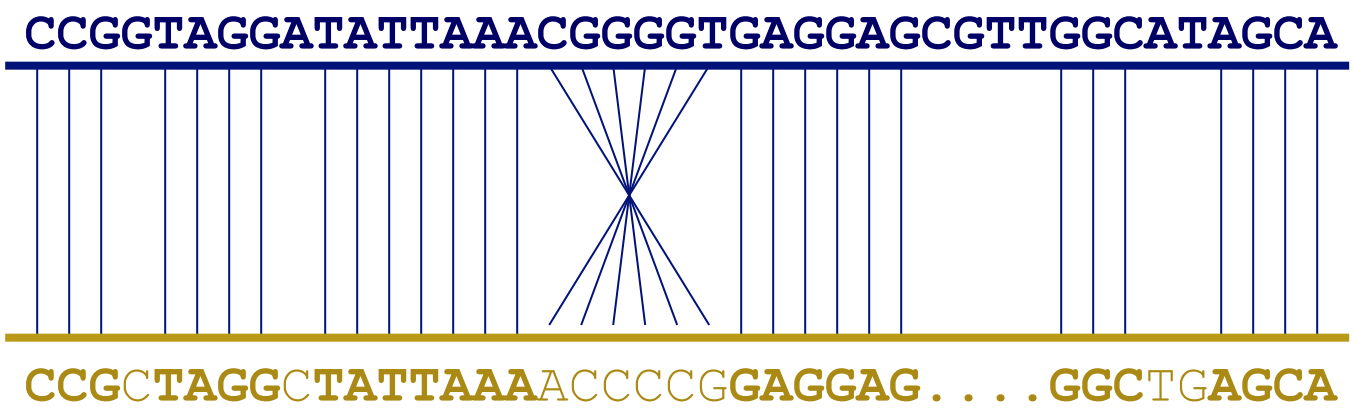
Goal of WGA

- For two genomes, *A* and *B*, find a mapping from each position in *A* to its corresponding position in *B*



Not so fast...

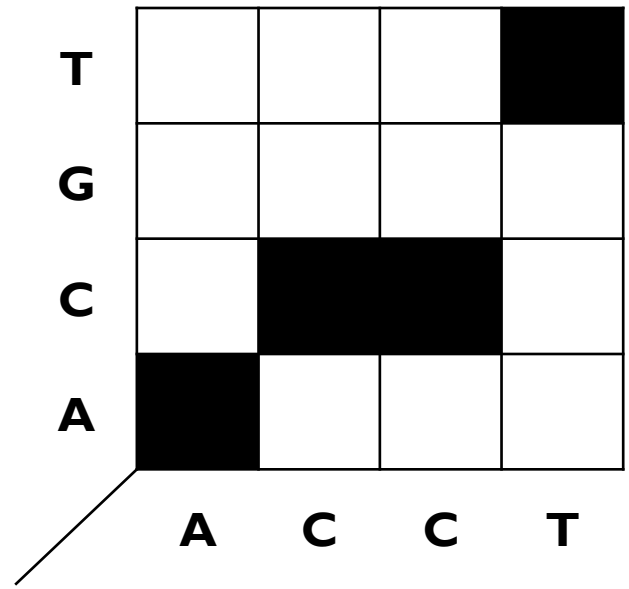
- Genome *A* may have insertions, deletions, translocations, inversions, duplications or SNPs with respect to *B* (sometimes all of the above)



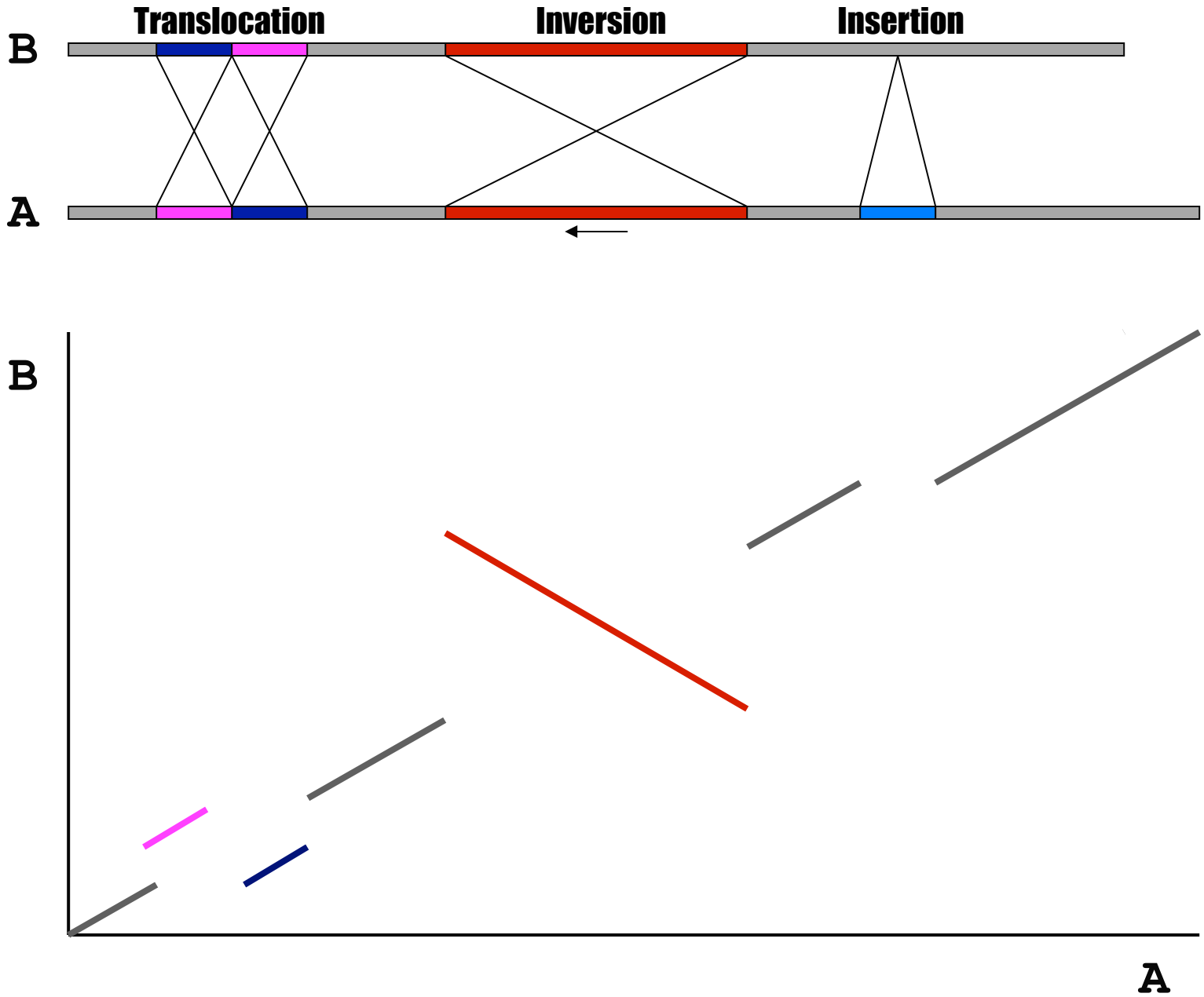
WGA visualization

- How can we visualize *whole* genome alignments?

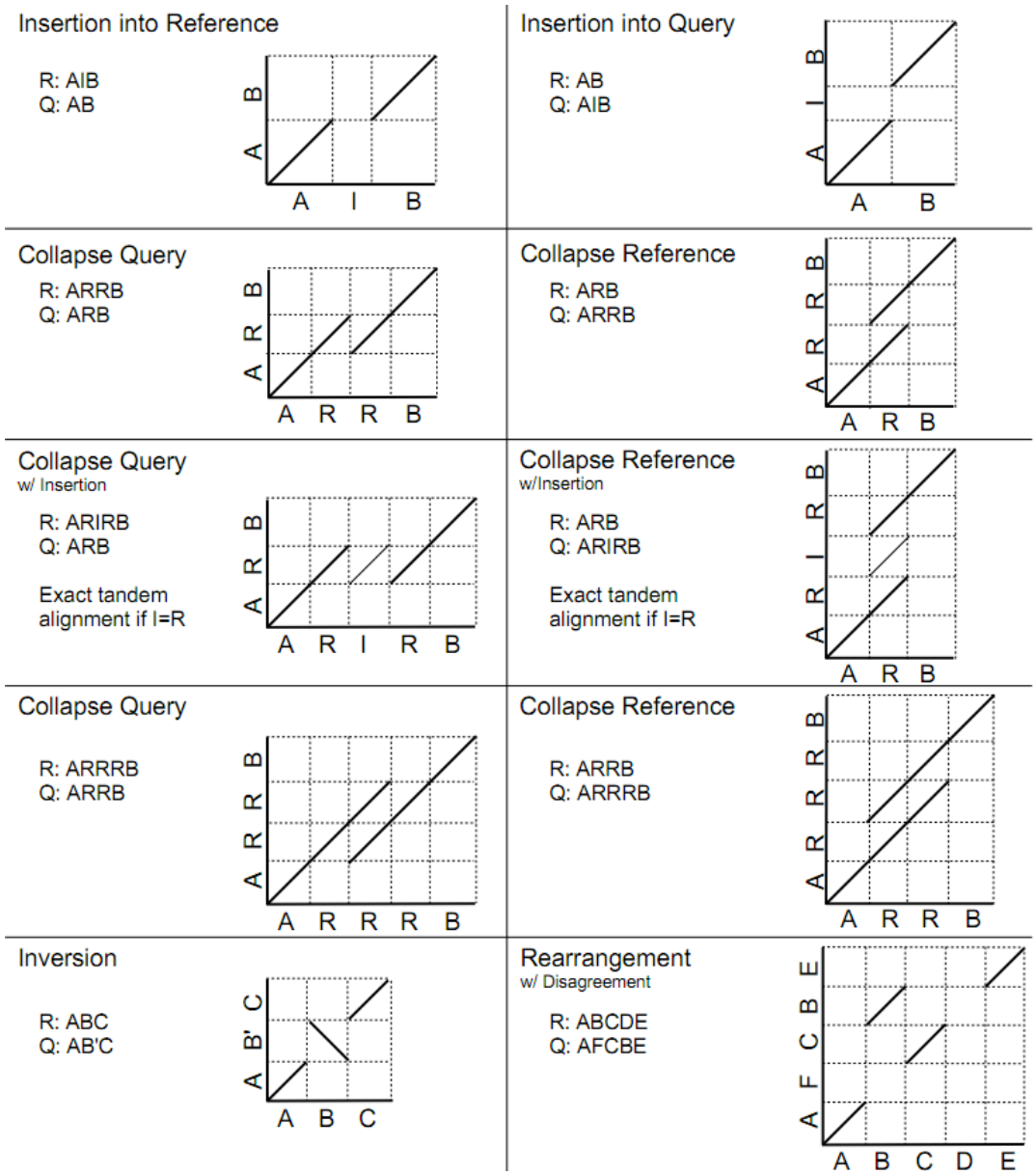
- With an alignment dot plot
  - *N* x *M* matrix
    - Let *i* = position in genome *A*
    - Let *j* = position in genome *B*
    - Fill cell *(i,j)* if A_i shows similarity to B_j



  - A perfect alignment between *A* and *B* would completely fill the positive diagonal


Translocation
Inversion
Insertion
B
A
B
A

SV Types



- Different structural variation types / misassemblies will be apparent by their pattern of breakpoints
- Most breakpoints will be at or near repeats
- Things quickly get complicated in real genomes

http://mummer.sf.net/manual/AlignmentTypes.pdf

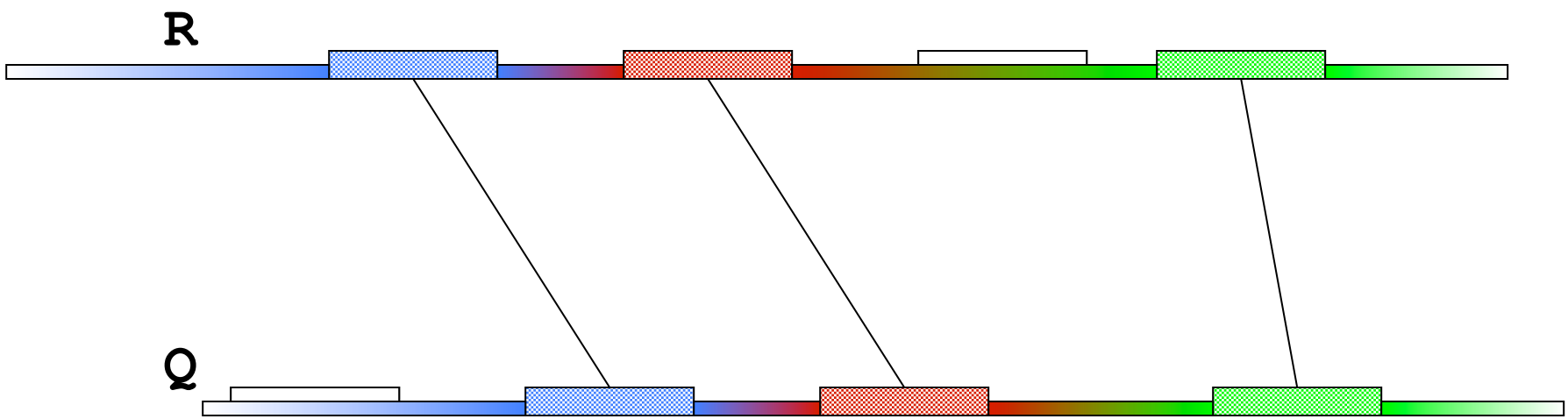
Seed and Extend

How can we quickly align two genomes?

FIND all exact matches (MUMs) of minimum length

CLUSTER consistent MUMs

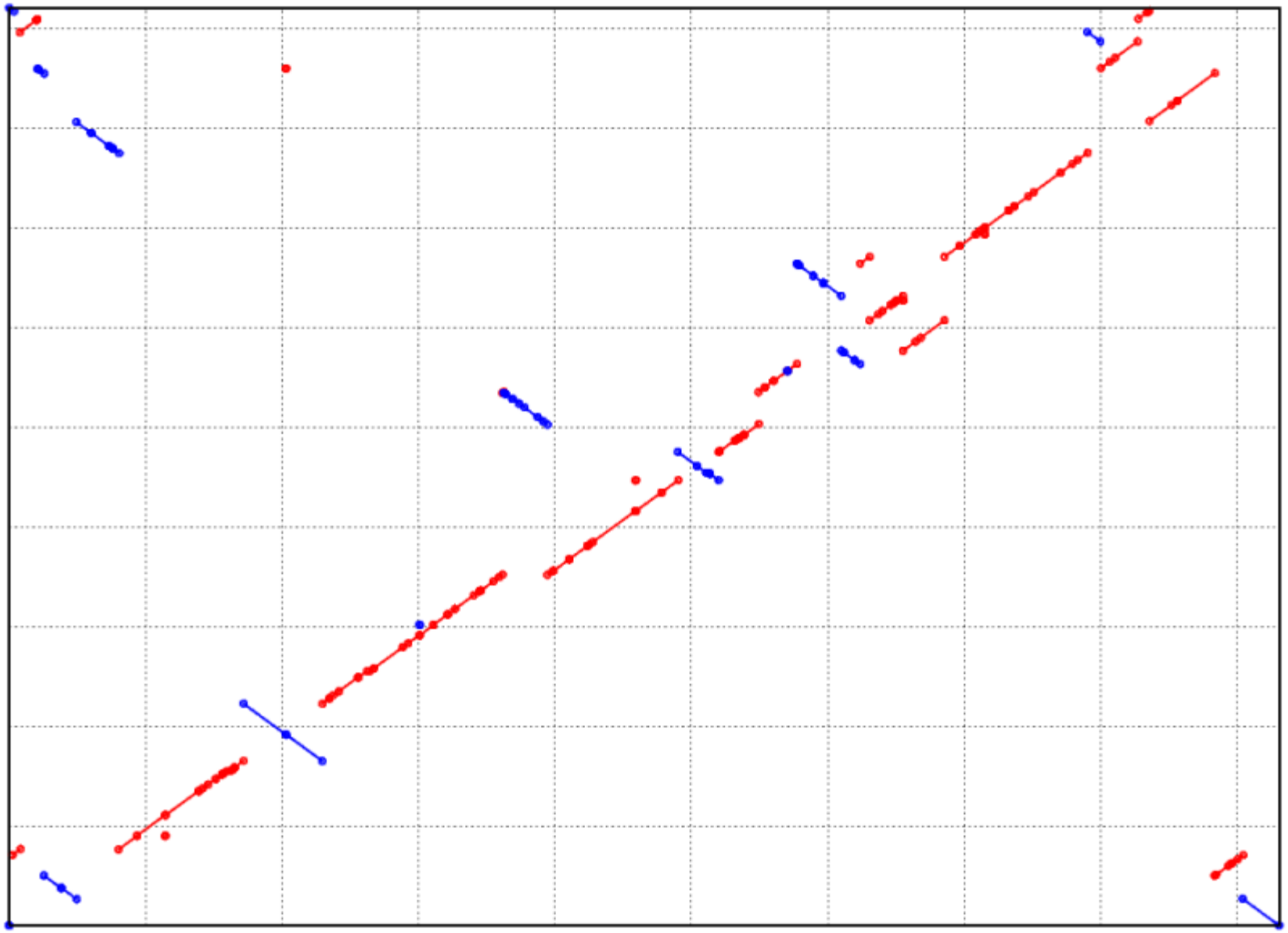
EXTEND alignments



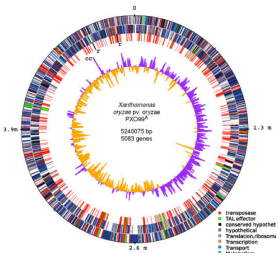
WGA example with **nucmer**

- *Yersina pestis CO92* vs. *Yersina pestis KIM*
  - High nucleotide similarity, 99.86%
  - Extensive genome shuffling
  - Highly repetitive

```
nucmer -maxmatch CO92.fasta KIM.fasta
-maxmatch    Find maximal exact matches (MEMs)

delta-filter -m out.delta > out.filter.m
-m  Many-to-many mapping

show-coords -r out.delta.m > out.coords
-r  Sort alignments by reference position

dnadiff out.delta.m
Construct catalog of sequence variations

mummerplot --layout out.delta.m
--layout Nice layout for multi-fasta files
```



Assembly Summary



Assembly quality depends on

1. ***Coverage***: low coverage is mathematically hopeless
2. ***Repeat composition***: high repeat content is challenging
3. ***Read length***: longer reads help resolve repeats
4. ***Error rate***: errors reduce coverage, obscure true overlaps

- Assembly is a hierarchical, starting from individual reads, build high confidence contigs/unitigs, incorporate the mates to build scaffolds
  - Extensive error correction is the key to getting the best assembly possible from a given data set

- Watch out for collapsed repeats & other misassemblies
  - Globally/Locally reassemble data from scratch with better parameters & stitch the 2 assemblies together

Thank You






http://schatzlab.cshl.edu/teaching/
@mike_schatz