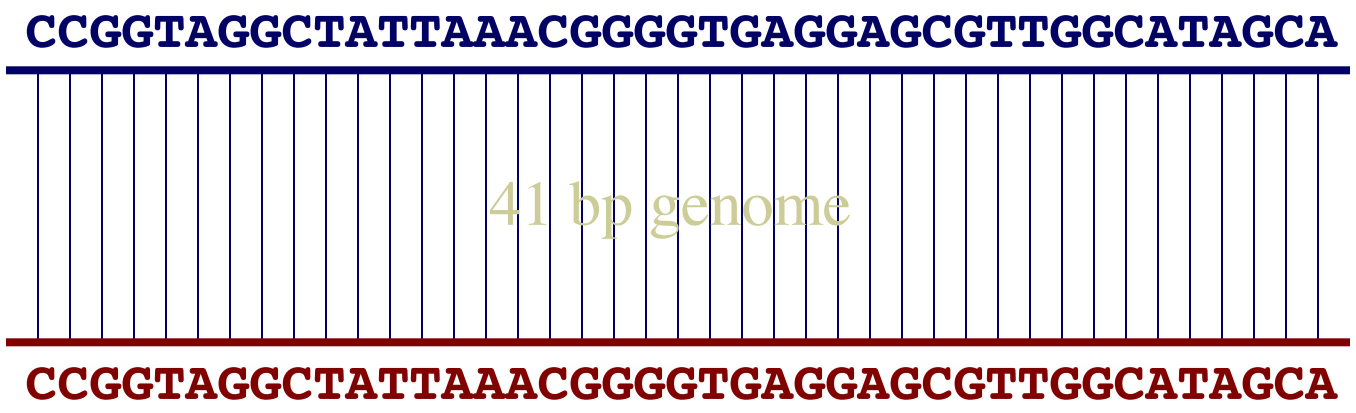
Whole Genome Alignment

TIGR Training Seminar
July 21th, 2006

Adam M Phillippy
Amp [at] cs.umd.edu

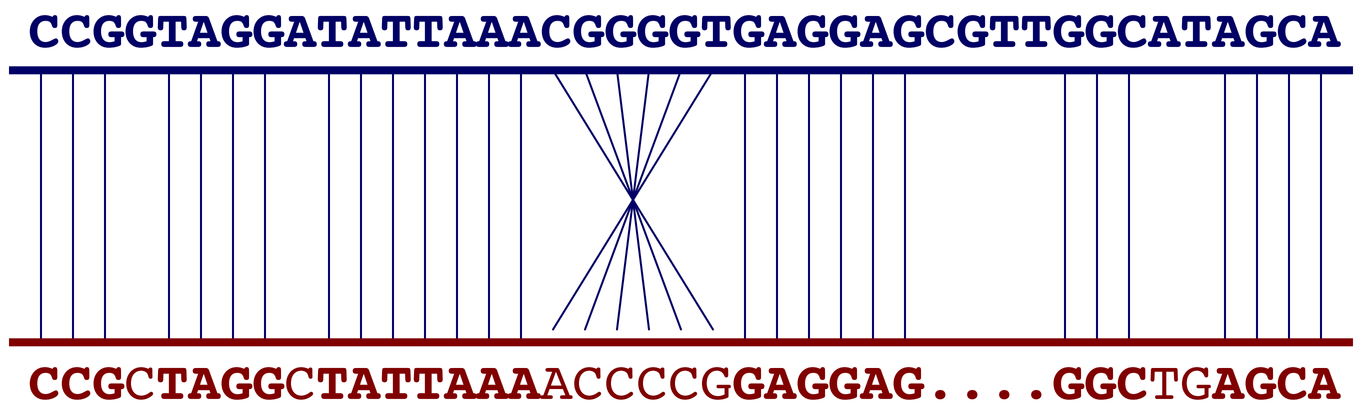
Goal of WGA

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


CCGGTAGGCTATTAAACGGGGTGAGGAGCGTTGGCATAGCA
41 bp genome
CCGGTAGGCTATTAAACGGGGTGAGGAGCGTTGGCATAGCA

Not so fast...

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



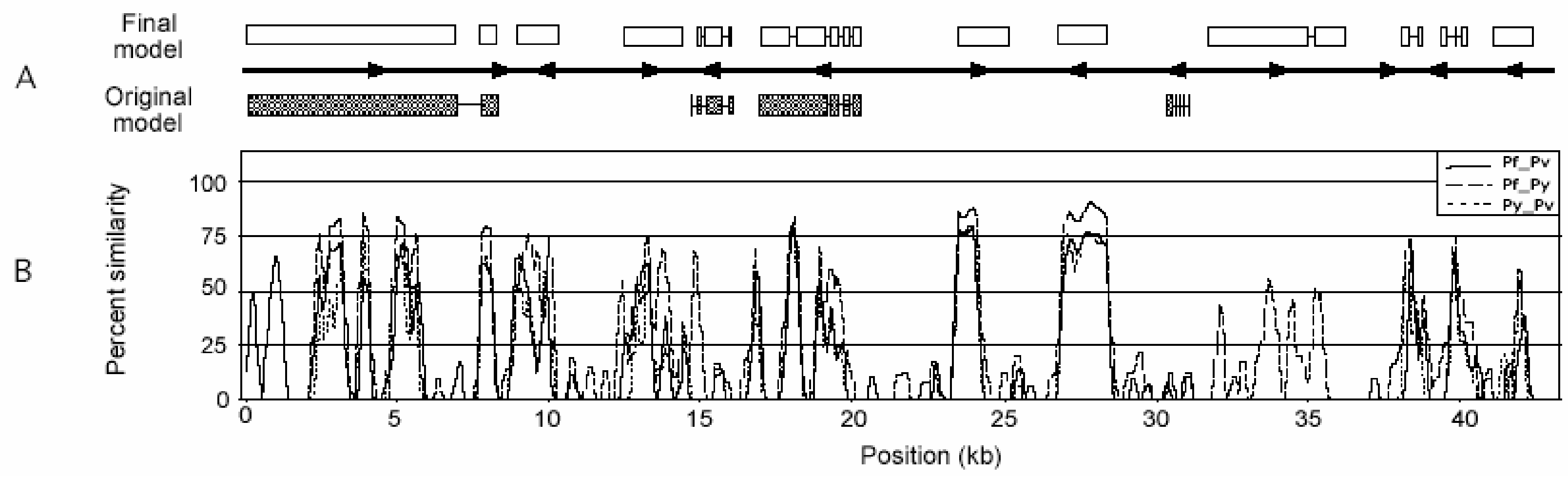
CCGGTAGGATATTAAACGGGGTGAGGAGCGTTGGCATAGCA
CCGCTAGGCTATTAAAACCCCGGAGGAG....GGCTGAGCA

Sidetrack: Plots

- How can we visualize alignments?

- With an identity plot
  - XY plot
    - Let x = position in genome *A*
    - Let y = %similarity of A_x to corresponding position in *B*
  - Plot the identity function
  - This can reveal islands of conservation, e.g. exons

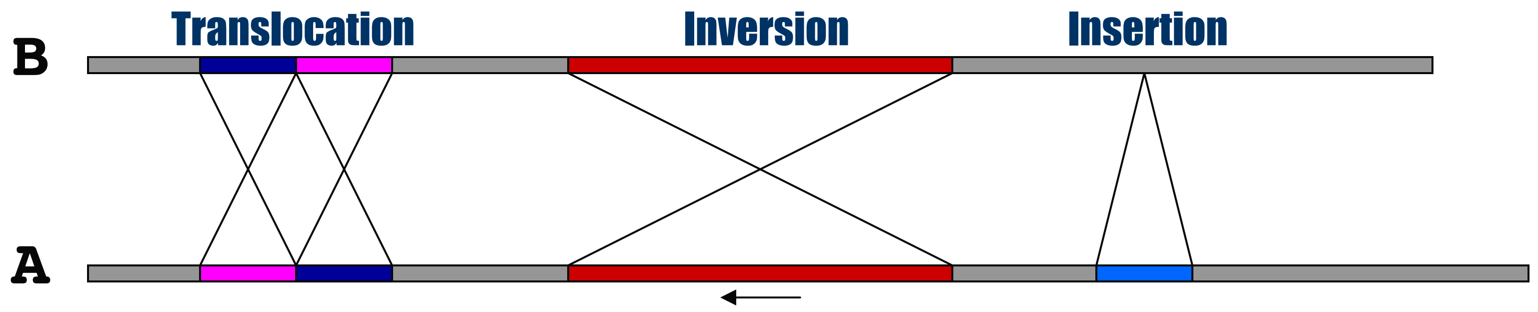
Identity plot example

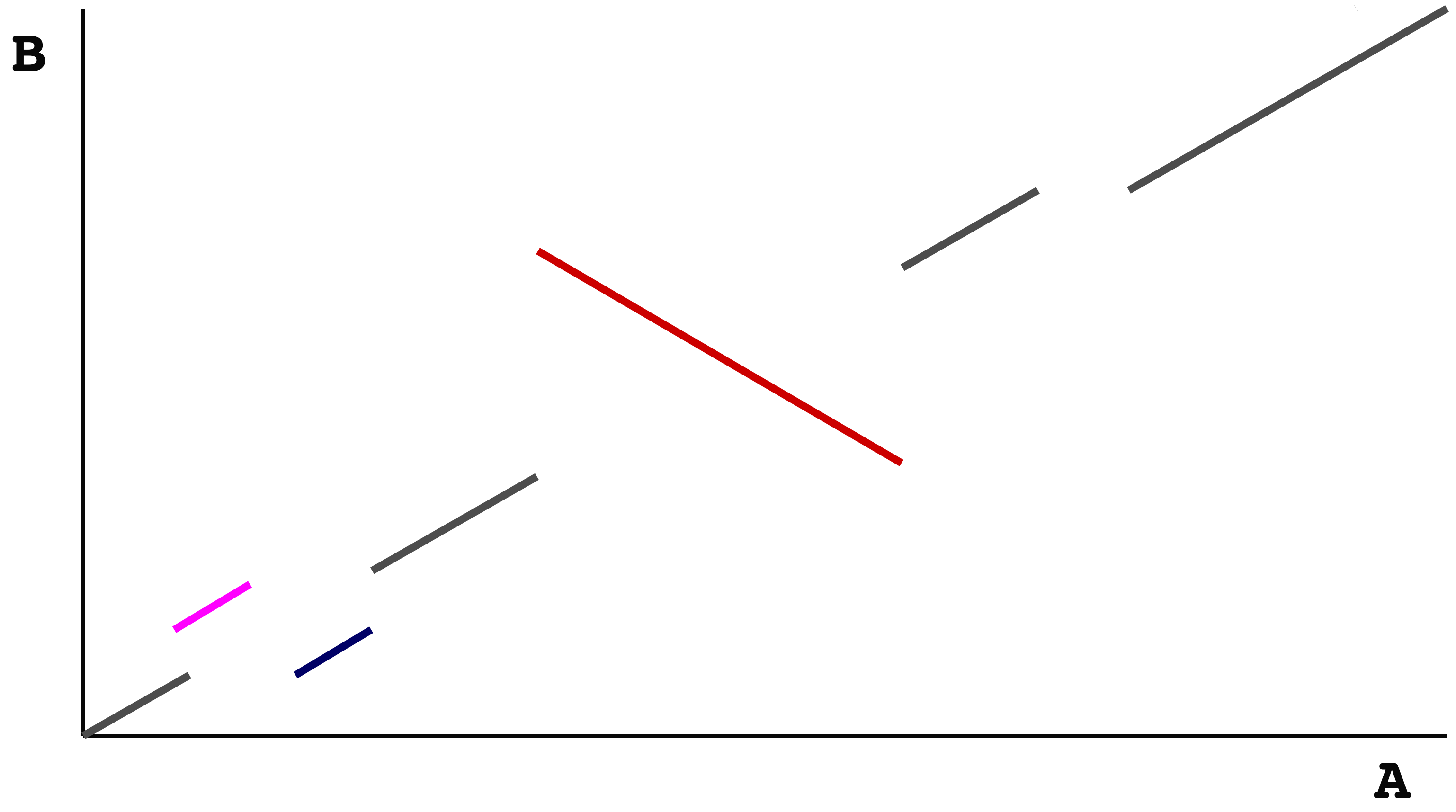


Sidetrack: Plots

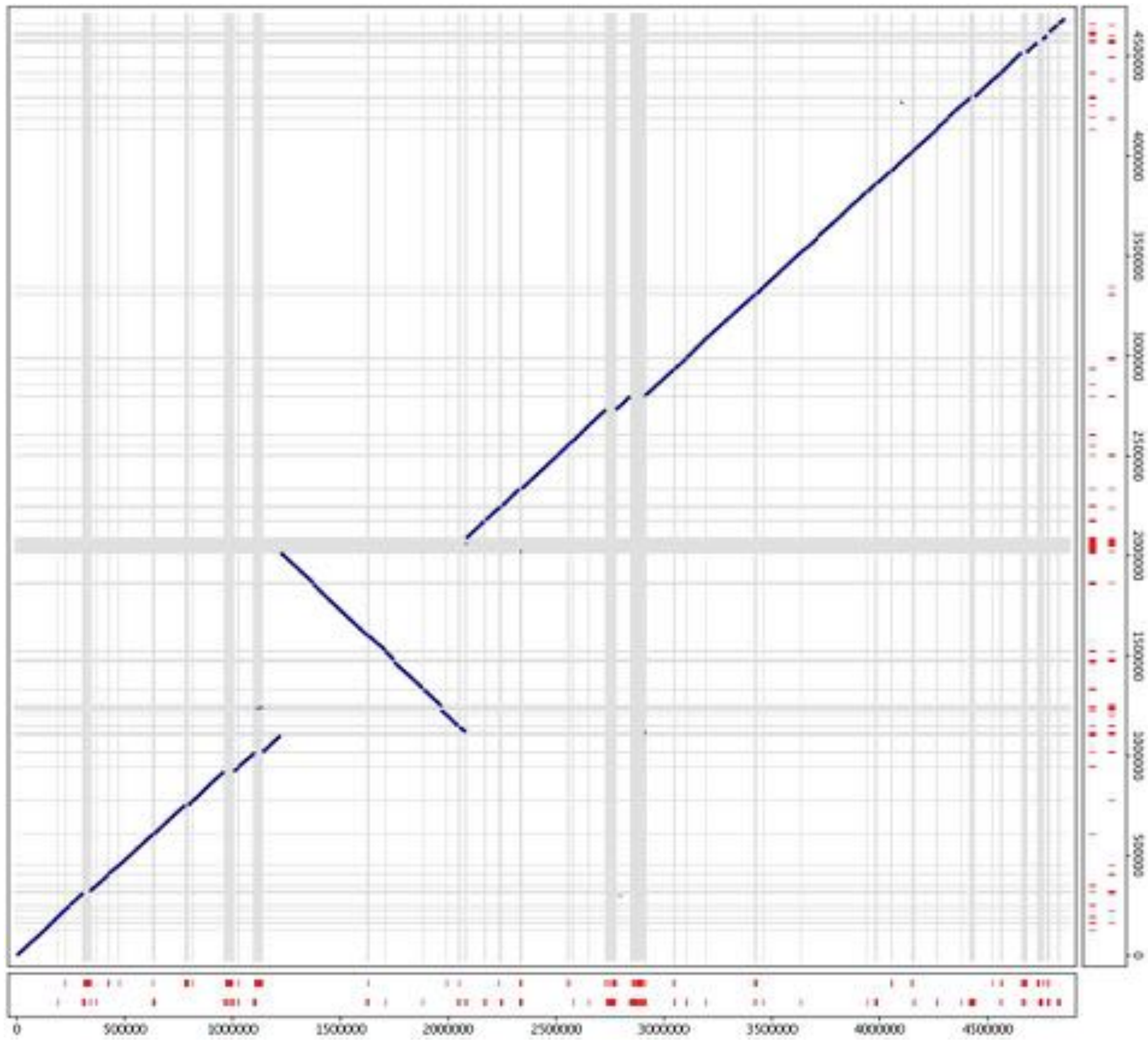
- How can we visualize *whole* genome alignments?

- 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





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


0
500000
1000000
1500000
2000000
2500000
3000000
3500000
4000000
4500000

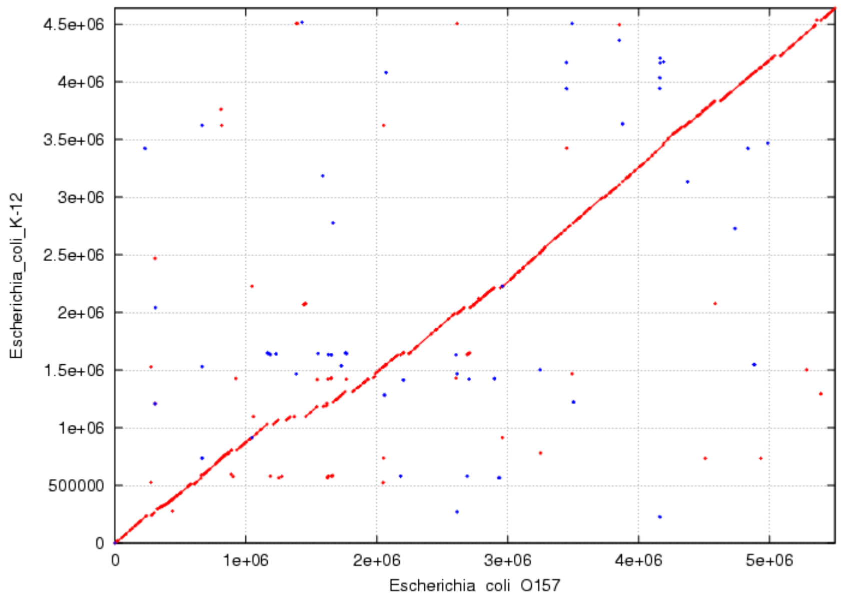
Global vs. Local

- Global pairwise alignment

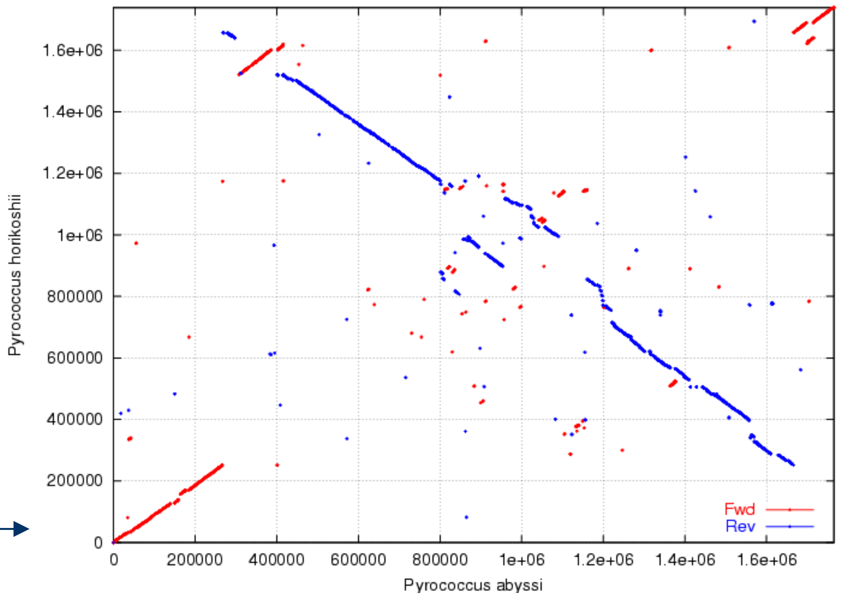
```
...AAGCTTGGCTTAGCTGCTAGGGTAGGCTTGGG...
...AAGCTGGGCTTAGTTGCTAG..TAGGCTTTGG...
        ^       ^      ^^       ^
```

- Whole genome alignment
  - Often impossible to represent as a global alignment
  - We will assume a set of local alignments (g-local)
    - This works great for draft sequence

Global vs. Local



global ok



global no way

Alignment Uses

- **Whole genome alignment**
  - Synteny analysis
  - Polymorphism detection
  - Sequence mapping
- **Multiple genome alignment**
  - Identify conserved sequence, e.g. functional elements (annotation)
  - Polymorphism detection
- **Multiple alignment**
  - Phylogenetics
  - Protein domain/structure analysis
- **Local sequence alignment**
  - Identify a DNA or protein sequence (annotation)
  - Sensitive homology search
  - Anchor a whole genome alignment

Alignment Tools

- **Whole genome alignment**
  - MUMmer*
    - Developed, supported and available at TIGR
  - LAGAN*, AVID
    - VISTA identity plots
- **Multiple genome alignment**
  - MGA, MLAGAN*, DIALIGN, MAVID
- **Multiple alignment**
  - Muscle?, ClustalW*
- **Local sequence alignment**
  - BLAST*, FASTA,Vmatch

* open source

MUMmer

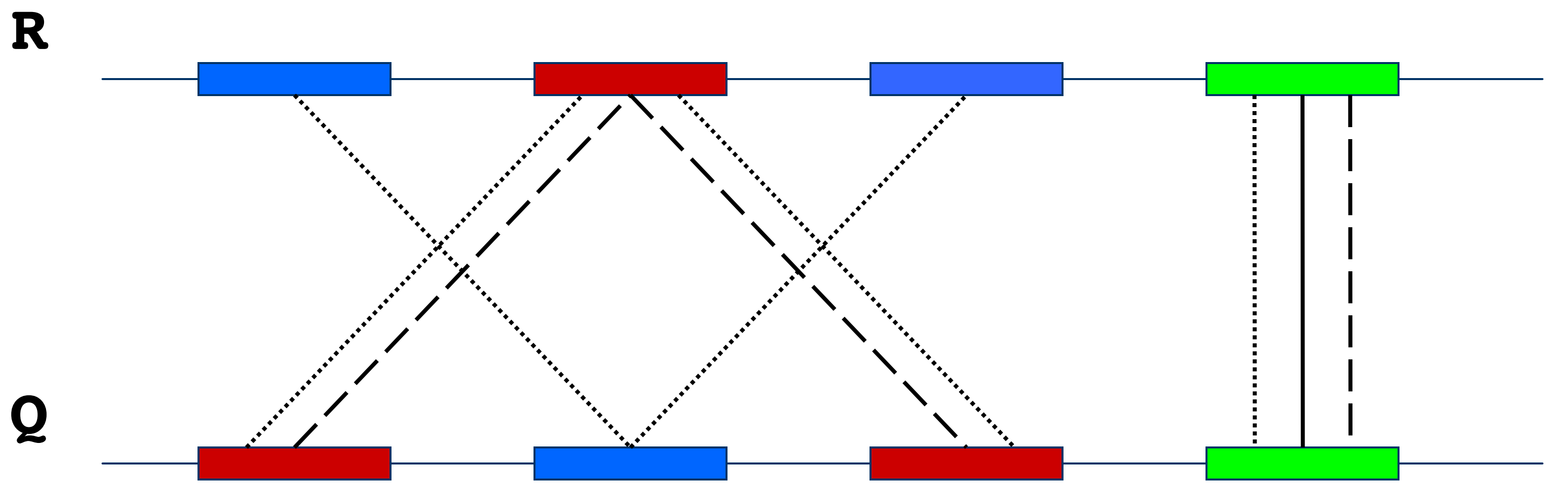
- <u>M</u>aximal <u>U</u>nique <u>M</u>atcher (MUM)
  - match
    - exact match of a minimum length
  - maximal
    - cannot be extended in either direction without a mismatch
  - *unique*
    - occurs only once in both sequences (MUM)
    - occurs only once in a single sequence (MAM)
    - occurs one or more times in either sequence (MEM)

Fee Fi Fo Fum,
is it a MAM, MEM or MUM?

MUM : maximal unique match ———

MAM : maximal almost-unique match – – – – –

MEM : maximal exact match ··········



Seed and Extend

- How can we make MUMs **BIGGER?**

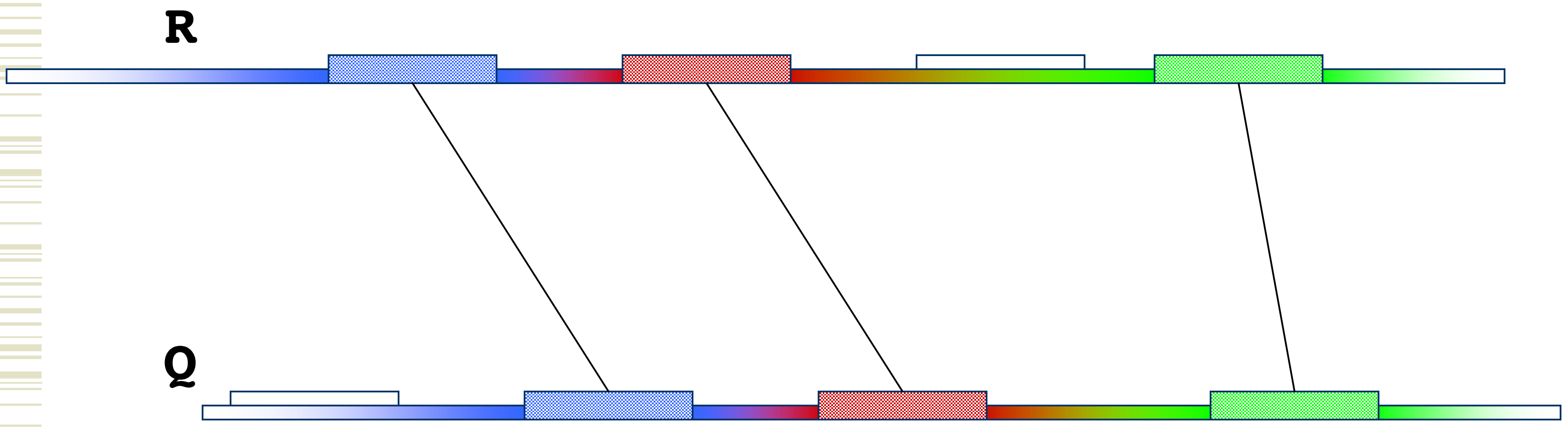
1. Find MUMs
   - using a suffix tree
2. Cluster MUMs
   - using size, gap and distance parameters
3. Extend clusters
   - using modified Smith-Waterman algorithm

Seed and Extend
visualization

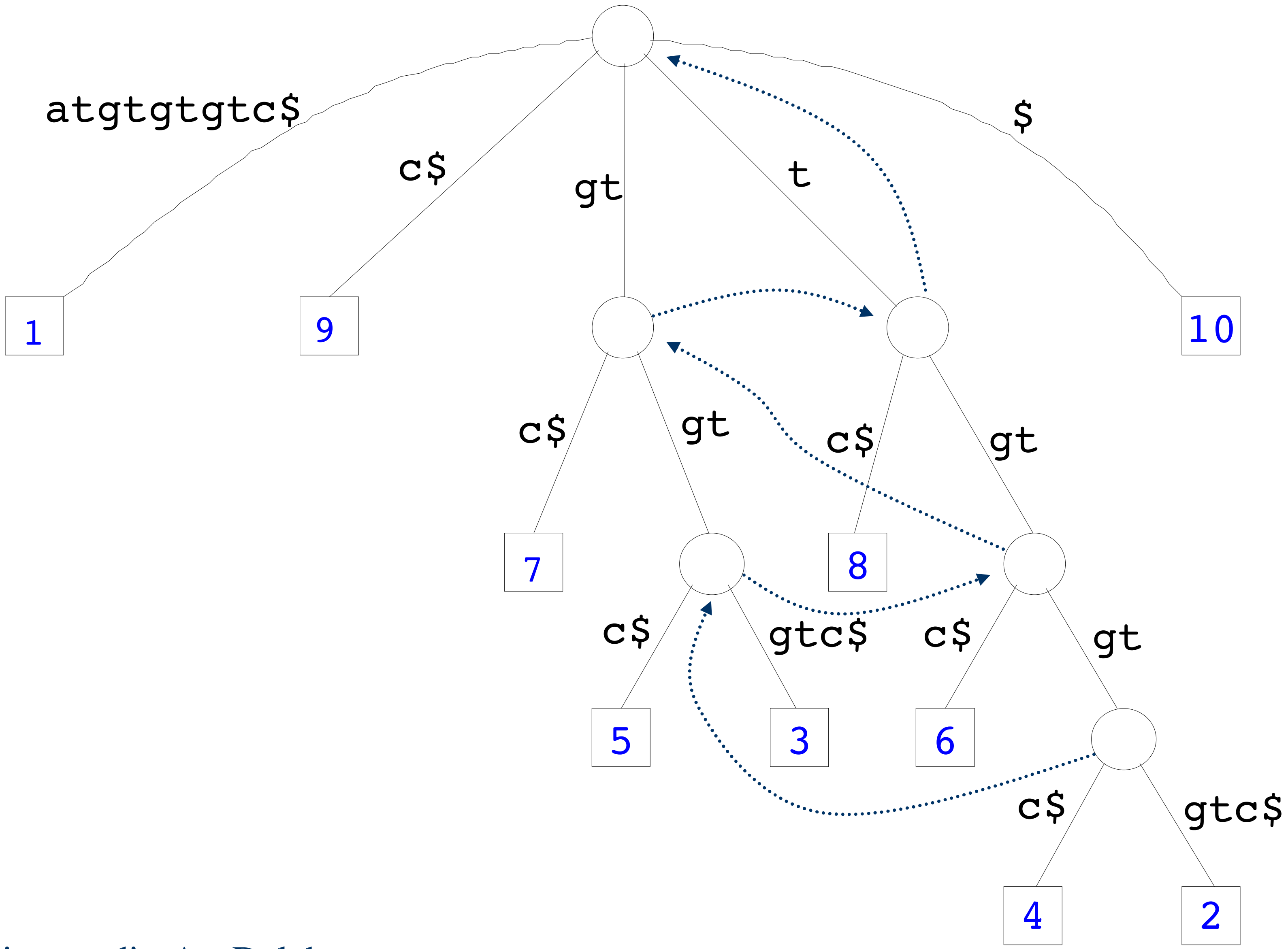
FIND all MUMs

CLUSTER consistent MUMs

EXTEND alignments



Suffix Tree for atgtgtgtc$



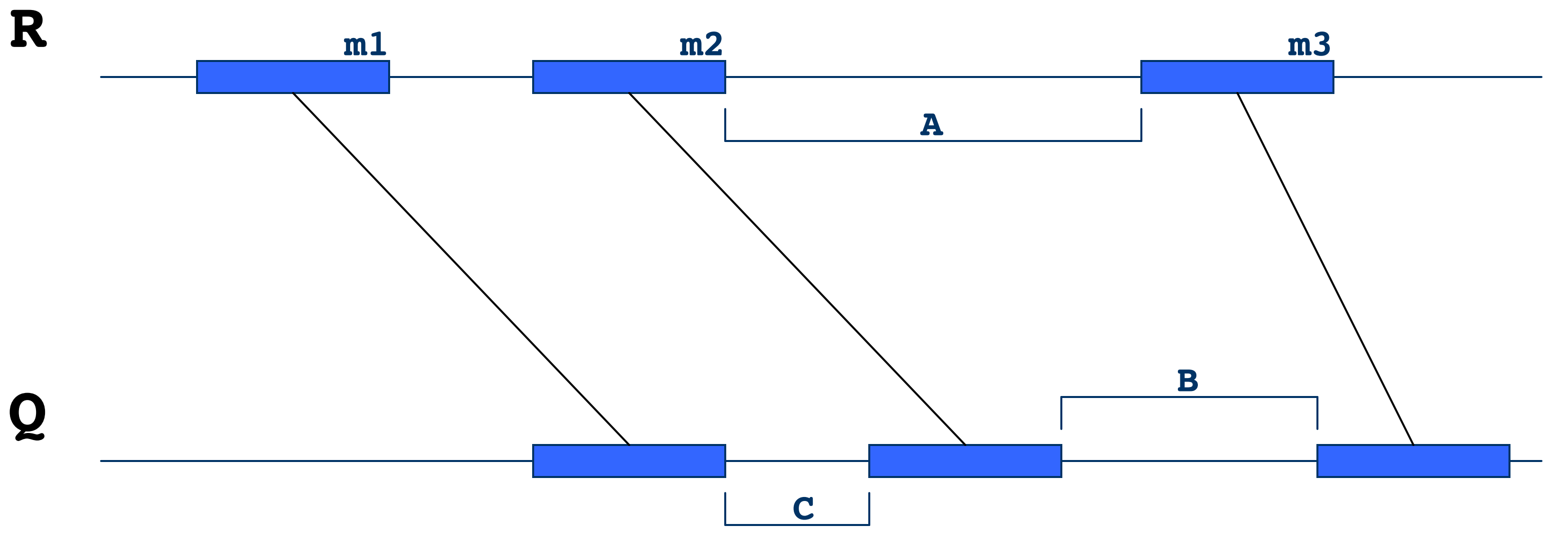
Drawing credit: Art Delcher

Clustering

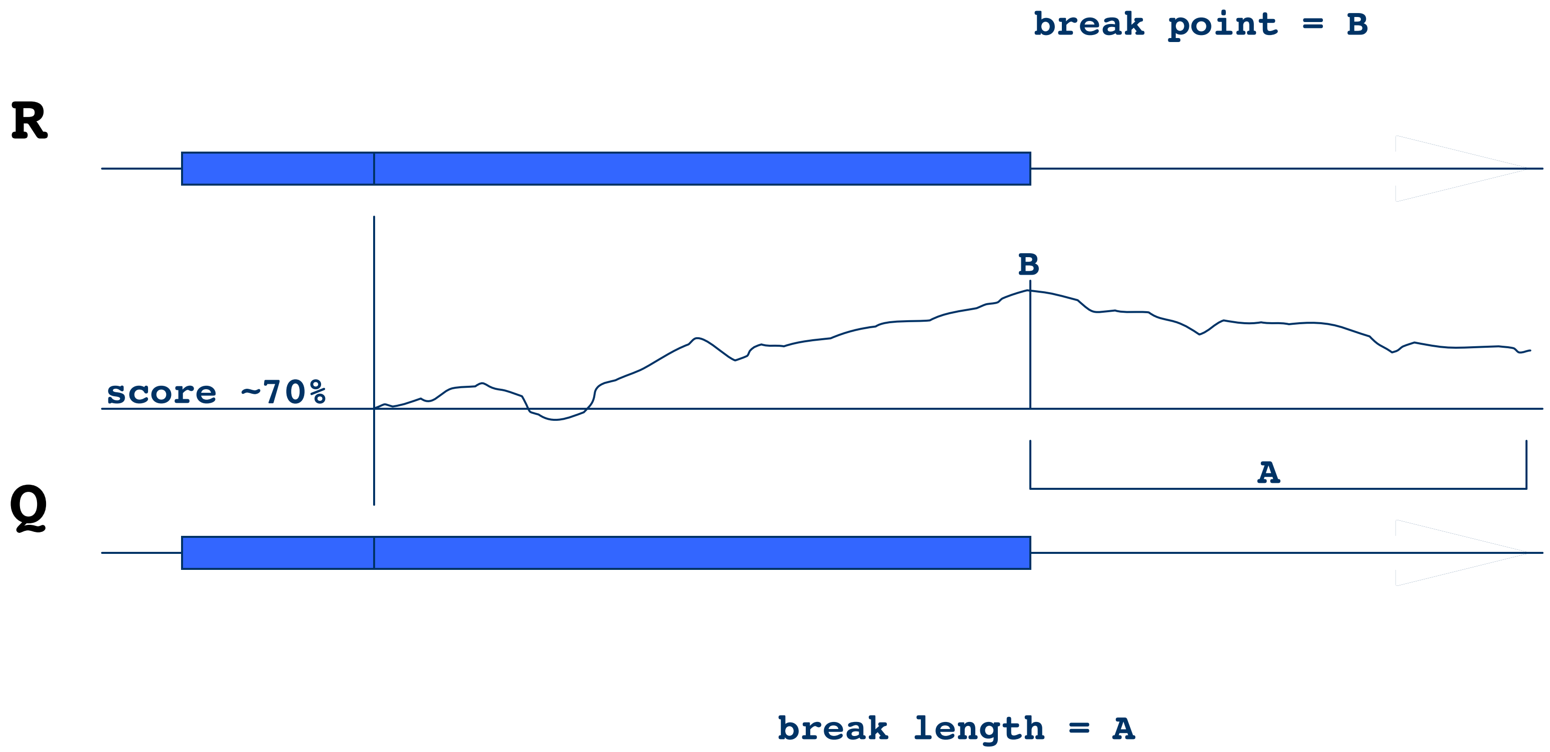
cluster length = Σm_i

gap distance = C

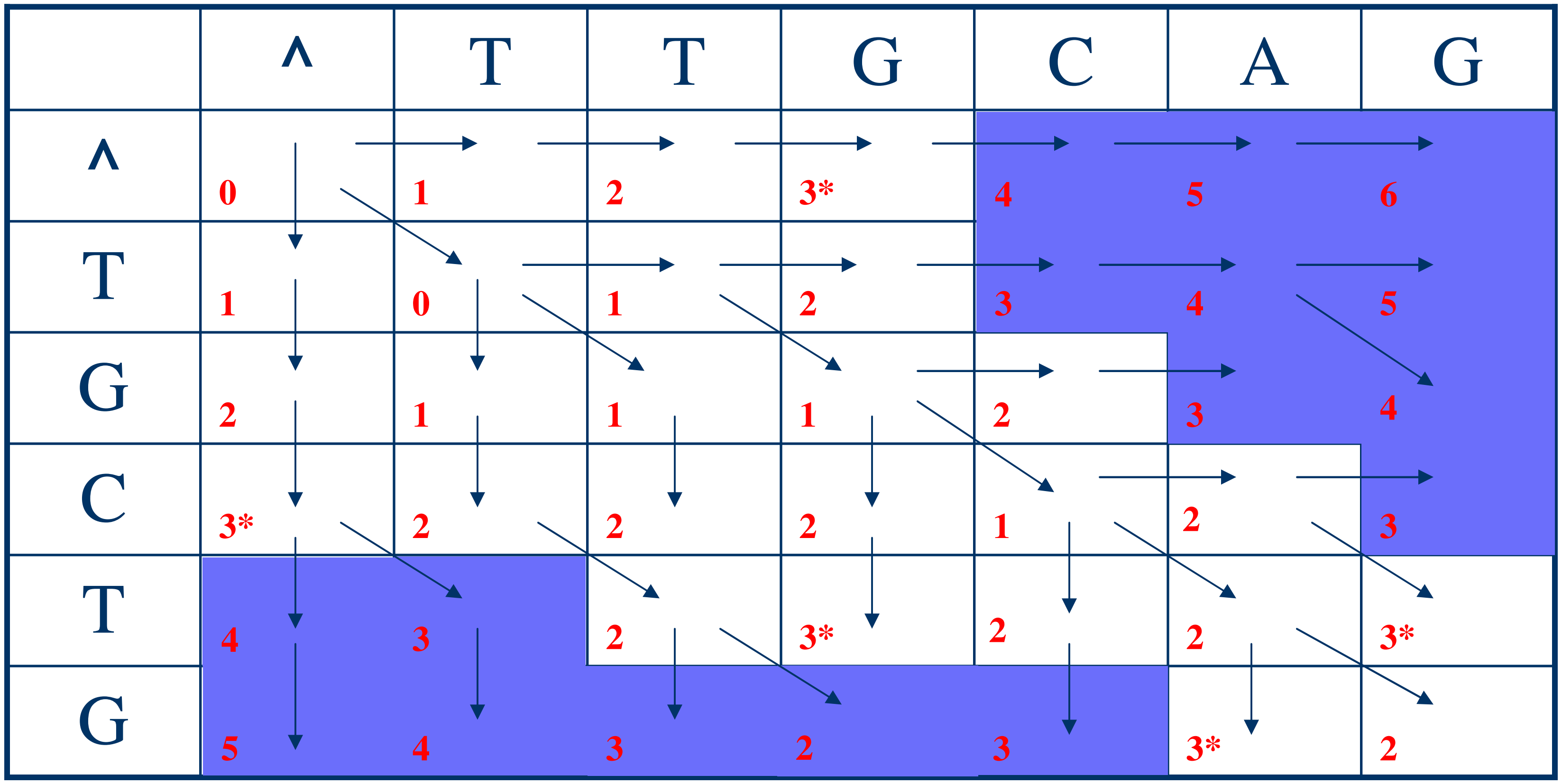
indel factor = $|B - A| / B$ *or* $|B - A|$



Extending



Banded Alignment



B

A

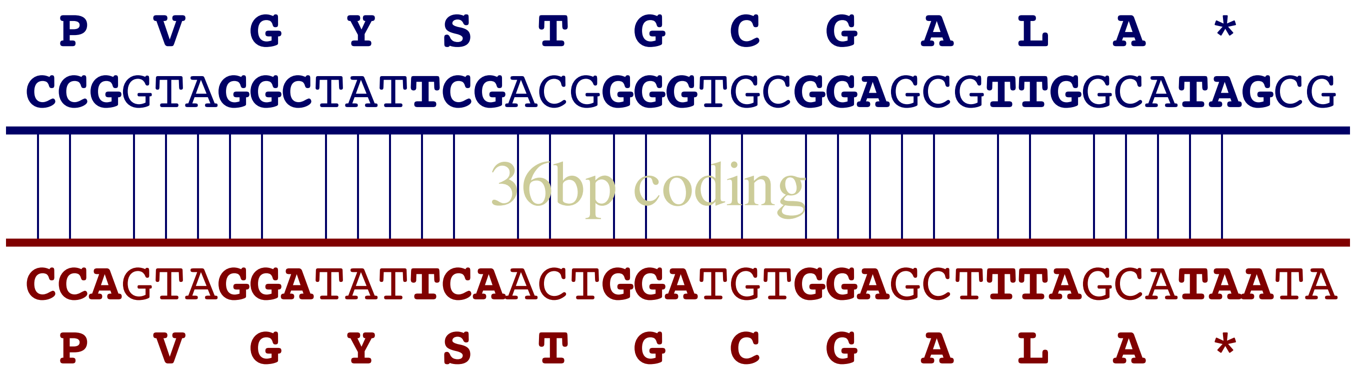
	^	T	T	G	C	A	G
^	0	1	2	3*	4	5	6
T	1	0	1	2	3	4	5
G	2	1	1	1	2	3	4
C	3*	2	2	2	1	2	3
T	4	3	2	3*	2	2	3*
G	5	4	3	2	3	3*	2

Adjustables

	nuc/promer options
Matching	
match length	-l
mum, mam, mem	-mum, -mumreference, -maxmatch
Clustering	
cluster length	-c
gap distance	-g
indel factor	-d
Extending	
search length	-b
scoring matrix	-x

Seedless Genes

- Single base pair substitution
  - non-synonymous mutation
  - synonymous mutation
    - 80% AT *Plasmodium falciparum*
    - 55% AT *Plasmodium vivax*



P V G Y S T G C G A L A *
CCGGTAGGCTATTCGACGGGGTGCGGAGCGTTGGCATAGCG
36bp coding
CCAGTAGGATATTCAACTGGATGTGGAGCTTTAGCATAATA
P V G Y S T G C G A L A *

Sidetrack: MUMmer suite

- ***mummer***
  - exact matching
- ***nucmer***
  - DNA multi-FastA input
  - whole genome alignment
- ***promer***
  - DNA multi-FastA input
  - whole genome alignment
- ***run-mummer1****
  - FastA input
  - global alignment
- ***run-mummer3****
  - FastA input w/ draft
  - whole genome alignment
- ***exact-tandems***
  - FastA input
  - exact tandem repeats

- NUCmer / PROmer utilities
  - ***mapview****
    - alignment plotter
    - draft sequence mapping
  - ***delta-filter***
    - alignment filter
  - ***mummerplot***
    - dot plotter
  - ***show-aligns***
    - pairwise alignments
  - ***show-coords***
    - alignment summary
  - ***show-snps***
    - snp reporting
  - ***show-tiling****
    - draft sequence tiling
- System utilities
  - ***gnuplot***
  - ***xfig***

* outdated

`mummer`

- Primary uses
  - exact matching (seeding)
  - dot plotting
- Pros
  - very efficient *O*(n) time and space
    - ~17 bytes per bp of reference sequence
    - *E. coli K12* vs. *E. coli O157:H7* (~5Mbp each)
      - 17 seconds using 77 MB RAM
  - multi-FastA input
- Cons
  - exact matches only

`nucmer` & `promer`

- Primary uses
  - whole genome alignment and analysis
  - draft sequence alignment
- Pros
  - multi-FastA inputs
  - well suited for genome and contig mapping
  - convenient helper utilities
    - `show-coords`, `show-snps`, `show-aligns`
    - `mummerplot`
- Cons
  - low sensitivity (w\ default parameters) with respect to BLAST

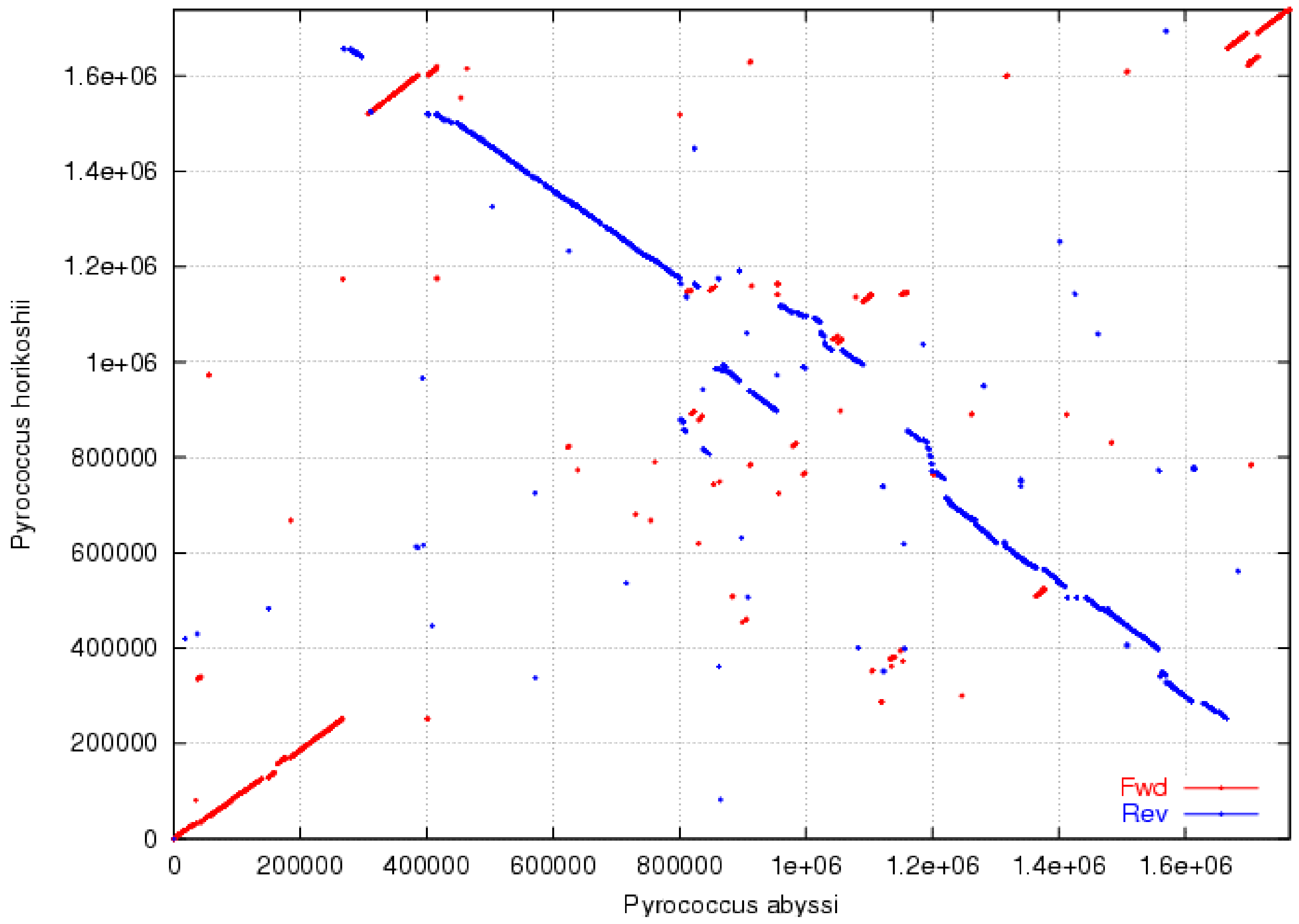
Applied MUMing

- Comparative genomics
  - dot plotting
  - synteny analysis
  - SNP detection
- Genome sequencing
  - draft sequence comparision
  - comparative scaffolding
  - contig and BAC overlaps
- Repeat detection
  - genomic repeats

WGA Example

- *Pyrococcus abyssi* vs. *horikoshii*
  - Hyperthermophilic Archaea
    - 100 °C / 200 bar
  - ~1.7 Mbp circular chromosome
  - ~58% unique genes at time of publication (1998)
  - Chromosome shuffling
    - "Pyrococcus genome comparison evidences chromosome shuffling-driven evolution." Zivanovic Y, Lopez Philippe, Philippe H, Forterre P, *Nucleic Acids Res*. 2002 May 1;30(9):1902-10.
  - See DAGchainer (B. Hass, *et al*.)
    - *Arabidopsis thaliana* segmental duplications

dotplot from promer-based mummerplot



COMMAND
dotplot

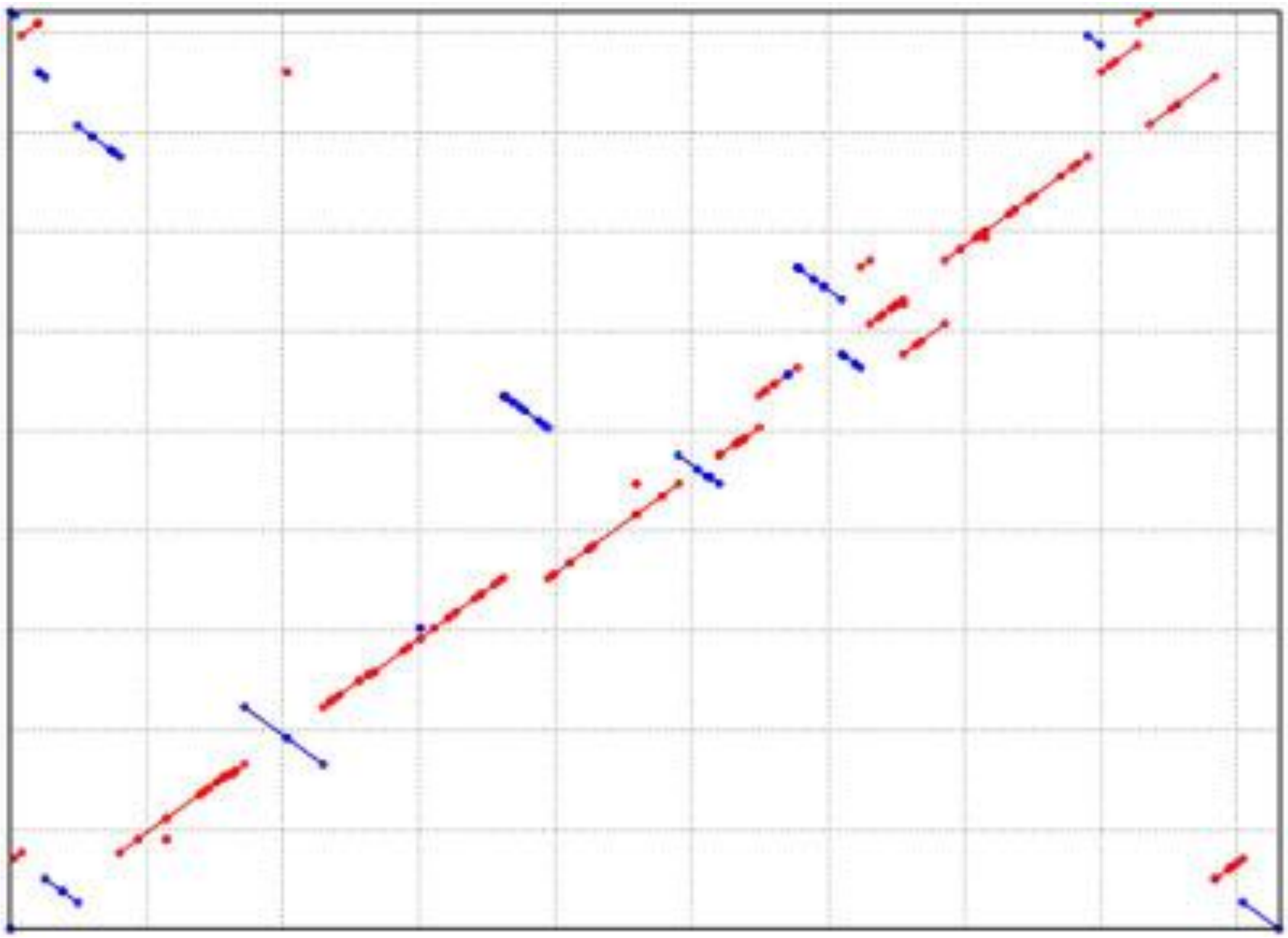
```
promer –mum –l 5 PABY.fasta PHOR.fasta
-mum                 Find maximal unique matches (MUMs)
-l                   Minimum match length (amino acids)

mummerplot –postscript out.delta
-postscript          Generate a postscript format plot
```

OR

```
mummer –mum –l 20 –b –c PABY.fasta PHOR.fasta > out.mums
mummerplot out.mums
```

SNP Example

- *Yersina pestis CO92* vs. *Yersina pestis KIM*
  - High nucleotide similarity, 99.86%
  - Extensive genome shuffling
    - Global alignment will not work
  - Highly repetitive
    - Will confuse local alignment (e.g. BLAST)



COMMAND
SNP detection

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

delta-filter –r -q out.delta > out.filter
-r             Filter out repetitive reference alignments
-q             Filter out repetitive query alignment

show-snps –r –I –T –x 10 out.filter > out.snps
-r             Sort SNPs by reference position
-I             Do not output indels
-T             Tab delimited output
-x 10          Output 10bp context for each SNP
```

show-snps output

- **[P1]** position of the SNP in the reference
- **[SUB]** reference base
- **[SUB]** query base
- **[P2]** position of the SNP in the query
- **[BUFF]** distance to the nearest polymorphism
- **[DIST]** distance to the nearest end of sequence
- **[R]** number of overlapping reference alignments (repeats)
- **[Q]** number of overlapping query alignments (repeats)
- **[LEN R]** length of the reference sequence
- **[LEN Q]** length of the query sequence
- **[CTX R]** context surrounding the reference base
- **[CTX Q]** context surrounding the query base
- **[FRM]** alignment orientation, 1 or -1 for forward or reverse
- **[TAGS]** the reference and query FastA IDs respectively

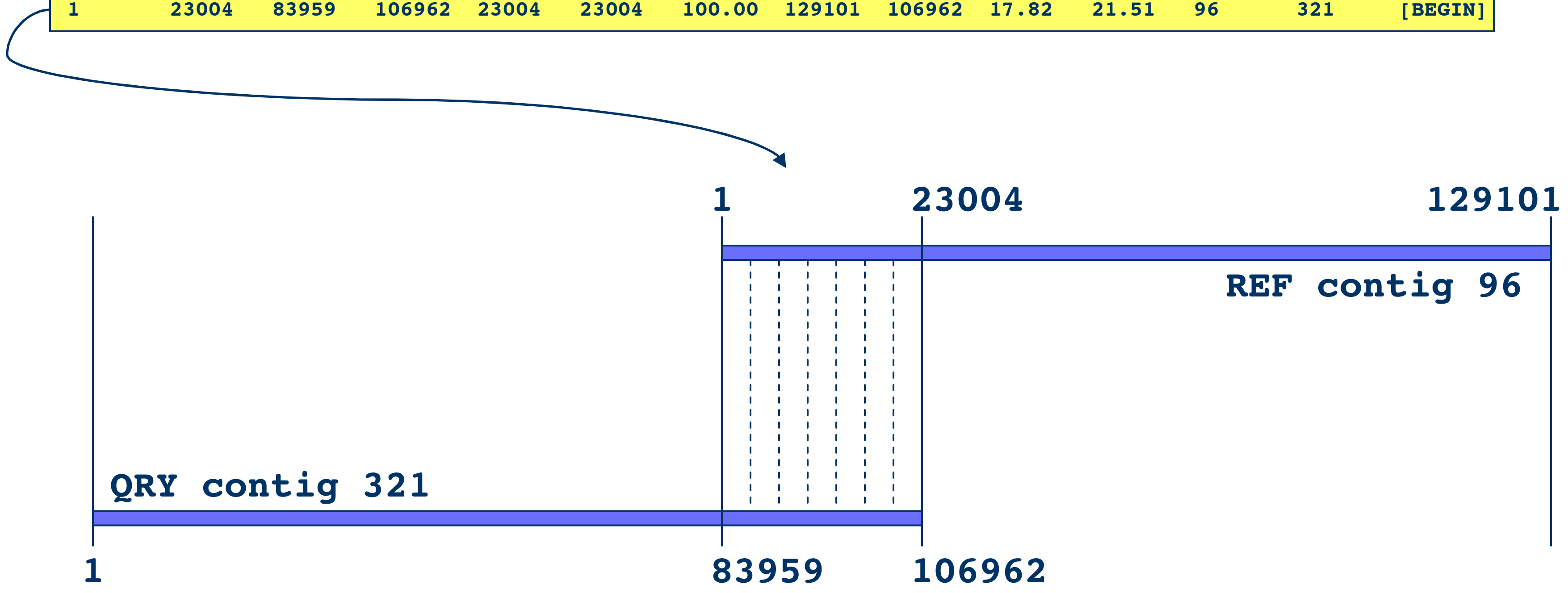
- All output coordinates and lengths are relative to the forward strand

COMMAND
BAC overlapping

```
nucmer –maxmatch BACS.fasta BACS.fasta
-maxmatch          Find maximal exact matches (MEMs)

show-coords –rcloT out.delta > out.coords
-r                 Sort alignments by reference
-c                 Display alignment coverage percentage
-l                 Display sequence length
-o                 Annotate overlaps between contigs
-T                 Tabular output

show-aligns –r out.delta REF_ID QRY_ID
-r                 Sort alignments by reference
```

BAC overlaps found by nucmer

```
[S1]     [E1]     [S2]     [E2]     [LEN 1]  [LEN 2]  [% IDY]  [LEN R]  [LEN Q]  [COV R]  [COV Q]  [TAGS]
==========================================================================================================
77793    127472   121884   72202    49680    49683    99.95    127472   121884   38.97    40.76    61       45       [END]
1        67053    56621    123672   67053    67052    99.91    127375   123672   52.64    54.22    72       18       [BEGIN]
1        111255   1        111255   111255   111255   99.99    111255   111255   100.00   100.00   74       75       [IDENTITY]
1        111255   1        111255   111255   111255   99.99    111255   111255   100.00   100.00   75       74       [IDENTITY]
107096   114214   116998   109898   7119     7101     98.08    114214   116998   6.23     6.07     76       332      [END]
55298    112695   1        57399    57398    57399    100.00   112695   130043   50.93    44.14    8        90       [END]
42551    116775   139969   65746    74225    74224    99.99    116775   139969   63.56    53.03    87       126      [END]
100319   101839   1        1521     1521     1521     99.41    125220   1521     1.21     100.00   89       561      [CONTAINS]
1        57399    55298    112695   57399    57398    100.00   130043   112695   44.14    50.93    90       8        [BEGIN]
1        23004    83959    106962   23004    23004    100.00   129101   106962   17.82    21.51    96       321      [BEGIN]
```



show-coords output

- **[S1]** start of the alignment region in the reference sequence
- **[E1]** end of the alignment region in the reference sequence
- **[S2]** start of the alignment region in the query sequence
- **[E2]** end of the alignment region in the query sequence
- **[LEN 1]** length of the alignment region in the reference sequence
- **[LEN 2]** length of the alignment region in the query sequence
- **[% IDY]** percent identity of the alignment
- **[% SIM]** percent similarity of the alignment
- **[% STP]** percent of stop codons in the alignment
- **[LEN R]** length of the reference sequence
- **[LEN Q]** length of the query sequence
- **[COV R]** percent alignment coverage in the reference sequence
- **[COV Q]** percent alignment coverage in the query sequence
- **[FRM]** reading frame for the reference and query sequence alignments respectively
- **[TAGS]** the reference and query FastA IDs respectively.

- All output coordinates and lengths are relative to the forward strand

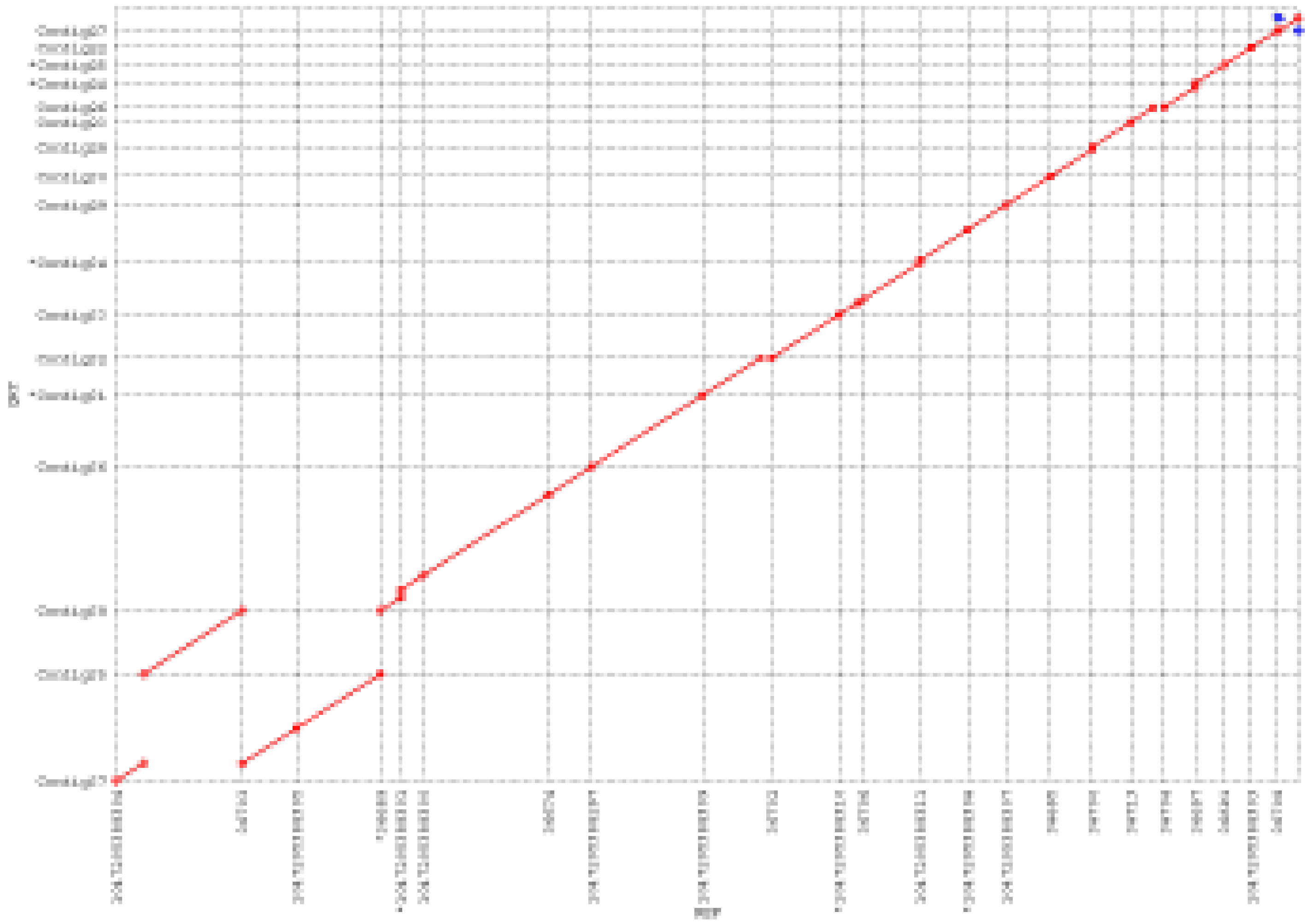
show-aligns output

```
-- BEGIN alignment [ +1 1 - 15407 | +1 1 - 15390 ]

1           agcttttcattctgactgcaacgggcaatatgtctctgtgtggattaaaaaaagagtctctgacagcagcttctgaactggttacctgc
1           agcttttcattctgactgcaacgggcaatatgtctctgtgtggattaaaaaaagagtgtctgatagcagcttctgaactggttacctgc
                                                                     ^     ^


90          cgtgagtaaattaaaattttattgacttaggtcactaaatactttaaccaatataggcatagcgcacagacagataaaaattacagagt
90          cgtgagtaaattaaaattttattgacttaggtcactaaatactttaaccaatataggcatagcgcacagacagataaaaattacagagt


179         acacaacatccatgaaacgcattagcaccaccattaccaccaccatcaccaccaccatcaccattaccattaccacaggtaacggtgcg
179         acacaacatccatgaaacgcattagcaccaccattaccaccaccatcacc..................attaccacaggtaacggtgcg
                                                              ^^^^^^^^^^^^^^^^^^


268         ggctgacgcgtacaggaaacacagaaaaaagcccgcacctgacagtgcgggcttttttt.tcgaccaaaggtaacgaggtaacaaccat
250         ggctgacgcgtacaggaaacacagaaaaaagcccgcacctgacagtgcgggcttttttttttcgaccaaaggtaacgaggtaacaaccat
                                                                       ^
```

COMMAND
draft sequence comparison

```
nucmer –maxmatch ASM1.fasta ASM2.fasta
-maxmatch         Use maximal exact matches (MEMs)

mummerplot –layout –large -filter out.delta
-layout           Permute alignment matrix for better viewing
-large            Big X11 (or postscript) plot
-filter           Auto-run ‘delta-filter –r –q’

   X11 Navigation:
   left-mouse: position
   middle-mouse: ruler
   right-mouse-drag: zoom-box
   N,P,U keys: next, previous, and un-zoom
```

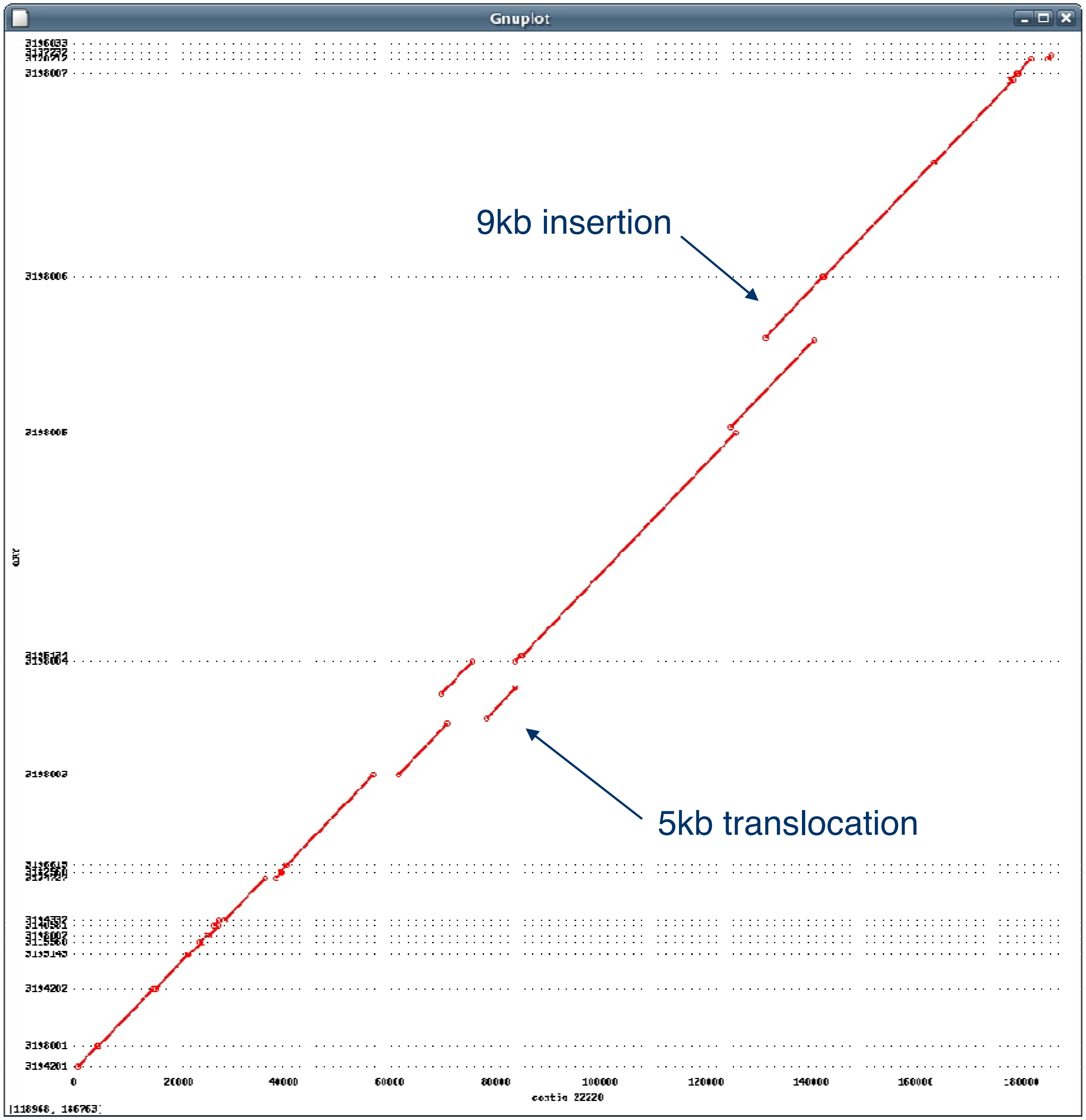
Multiple contig alignment by nucmer



Arachne vs. CA

D. virilis assemblies

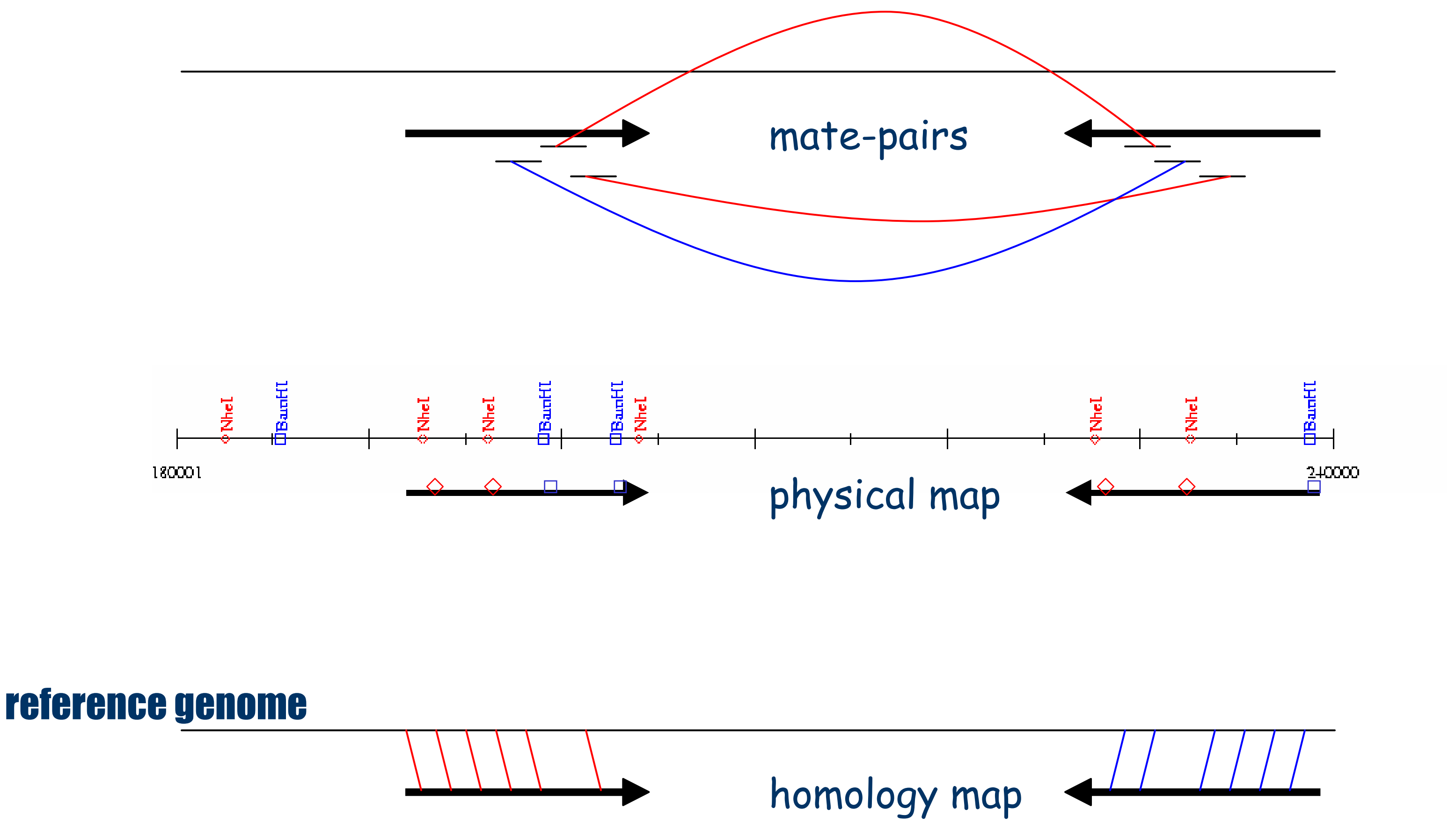
Arachne contig (X) mapping to multiple CA contigs (Y). Two macroscopic differences are highlighted, hundreds were found.



Comparative Scaffolding

- Scaffolding
  - order and orient draft contigs
    - using WGS mate-pair information
    - using physical map information
- Comparative Scaffolding
  - order and orient draft contigs
    - using a reference genome and alignment mapping
      - nucmer
  - very useful for physical gaps
  - can instantly close some sequencing gaps (overlapping contigs)

Comparative Scaffolding



COMMAND
contig mapping

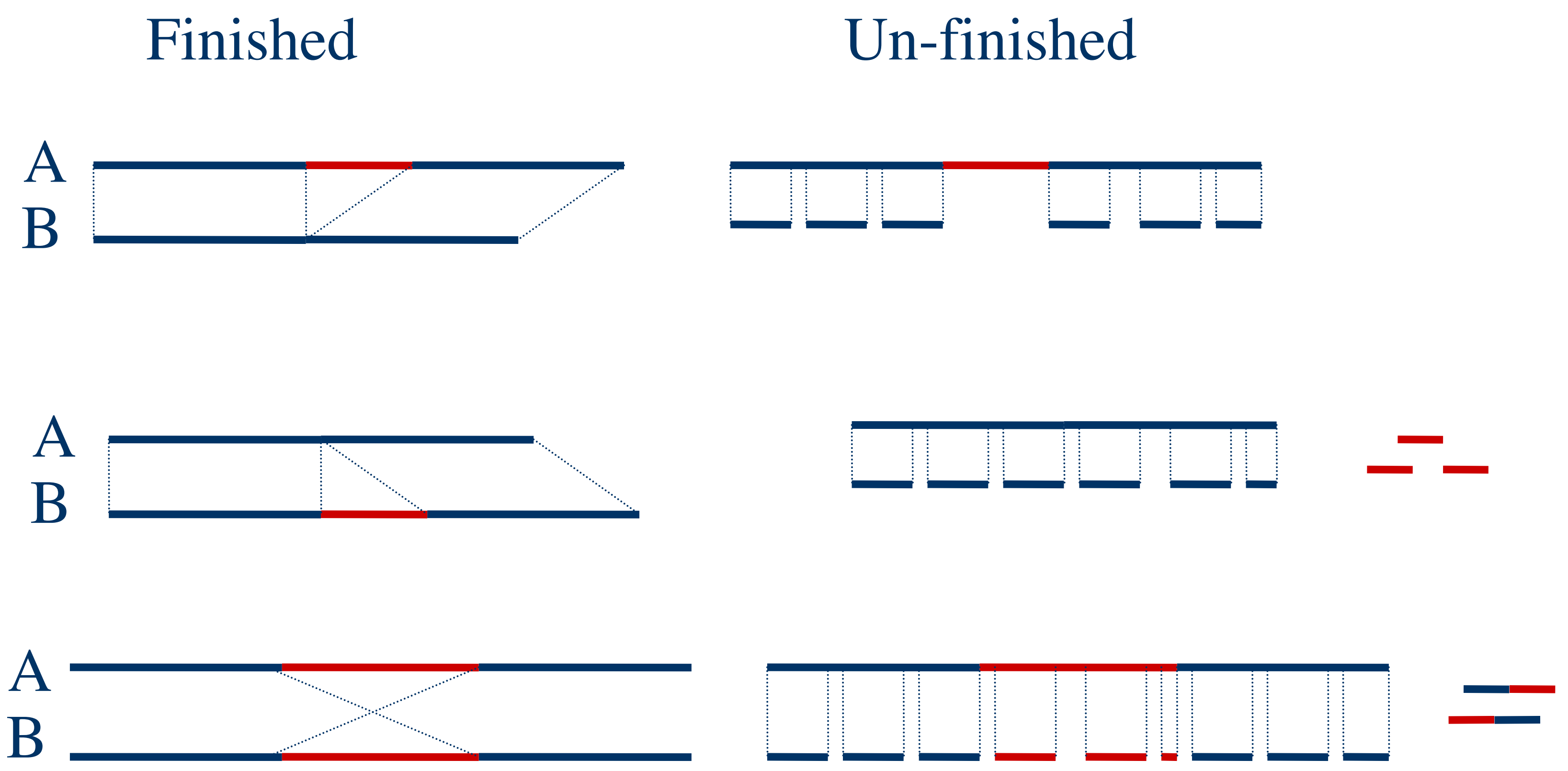
```
nucmer –maxmatch REF.fasta CTGS.fasta
-maxmatch     Find maximal exact matches (MEMs)

delta-filter –q out.delta > out.delta.filter
-q            Filter out repetitive query alignments

show-coords –rcl out.delta > out.coords
-r            Sort alignments by reference
-c            Display alignment coverage percentage
-l            Display sequence length
```

Read Mapping

- Comparative assembly
  - Neanderthal genome, NY Times
    - 454 pyrosequencing
      - 100bp reads
      - no mate-pairs

```
nucmer –maxmatch –l 15 –c 40
delta-filter –q
show-coords -q
```

Comparative Mapping
caveats

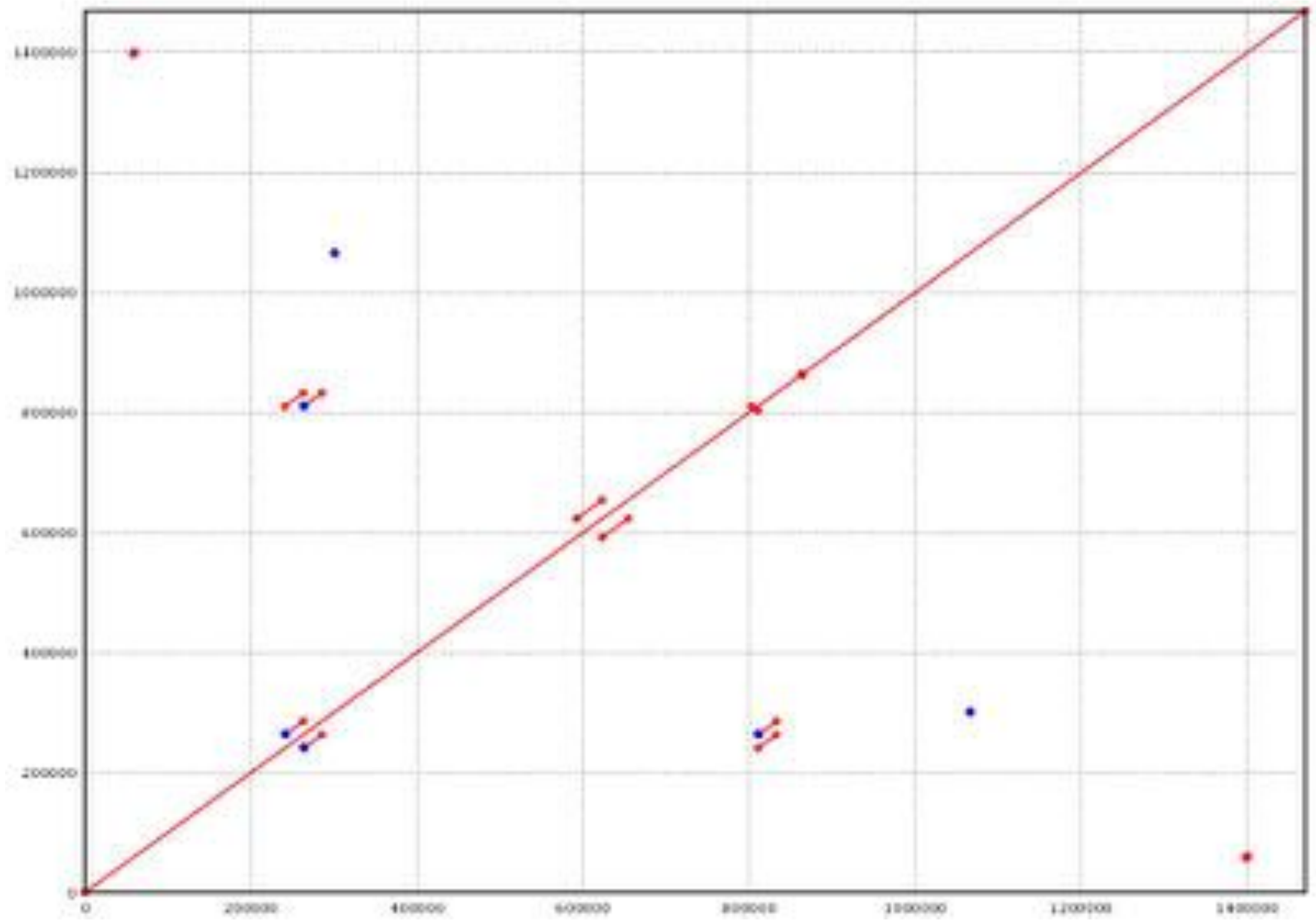


…RepeatsRepeatsRepeats…

- Exact repeats, palandromes, tandems, etc.
  - Use Vmatch
    - http://www.vmatch.de
- Long, inexact repeats
  - Use nucmer
    - genomic repeats `-maxmatch` **`-nosimplify`**
    - contig / BAC overlaps `-maxmatch`

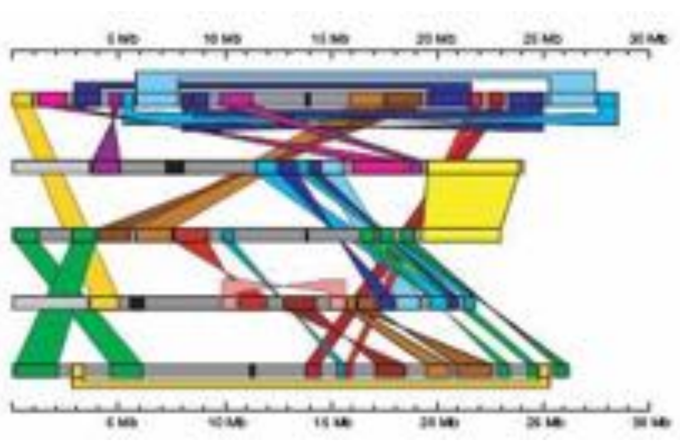
genomic repeats found by ‘nucmer --maxmatch --nosimplify’

```
    [S1]     [E1]  |     [S2]     [E2]  |  [LEN 1]  [LEN 2]  |  [% IDY]  | [TAGS]
=====================================================================================

   57832    60483  |  1398170  1400821  |     2652     2652  |    99.89  | gde:6876     gde:6876
  240759   242028  |   264386   263117  |     1270     1270  |   100.00  | gde:6876     gde:6876
  240759   263123  |   810529   832893  |    22365    22365  |    99.99  | gde:6876     gde:6876
  242022   263123  |   264380   285481  |    21102    21102  |    99.99  | gde:6876     gde:6876

  263117   264386  |   811798   810529  |     1270     1270  |   100.00  | gde:6876     gde:6876

  264380   285490  |   811792   832902  |    21111    21111  |    99.99  | gde:6876     gde:6876
  300630   301615  |  1066580  1065595  |      986      986  |    98.88  | gde:6876     gde:6876
  592225   623250  |   623236   654262  |    31026    31027  |    99.99  | gde:6876     gde:6876

  803061   803126  |   810475   810540  |       66       66  |   100.00  | gde:6876     gde:6876

  862678   863090  |   864053   864465  |      413      413  |    78.74  | gde:6876     gde:6876
```

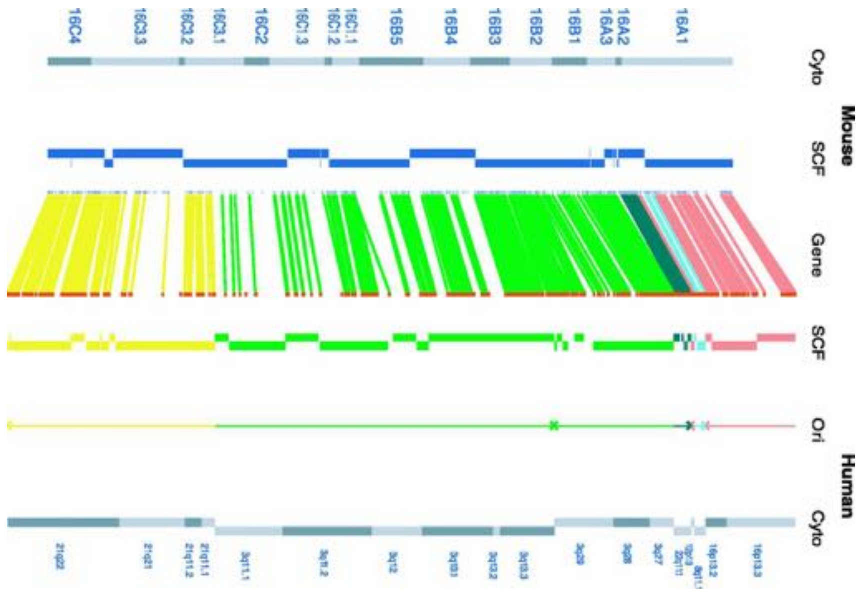

0
200000
400000
600000
800000
1000000
1200000
1400000
0
200000
400000
600000
800000
1000000
1200000
1400000

References

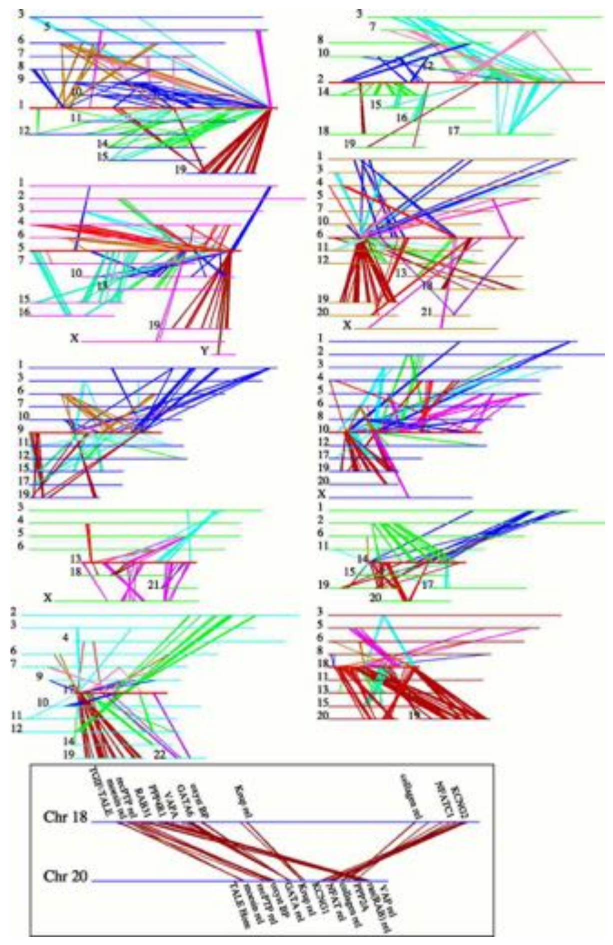
- Documentation
  - http://mummer.sourceforge.net
    - publication listing
  - http://mummer.sourceforge.net/manual
    - thorough documentation
  - http://mummer.sourceforge.net/examples
    - Walkthroughs
- Email
  - mummer-help (at) lists.sourceforge.net
  - mummer-users (at) lists.sourceforge.net



5 Mb
10 Mb
15 Mb
20 Mb
25 Mb
30 Mb



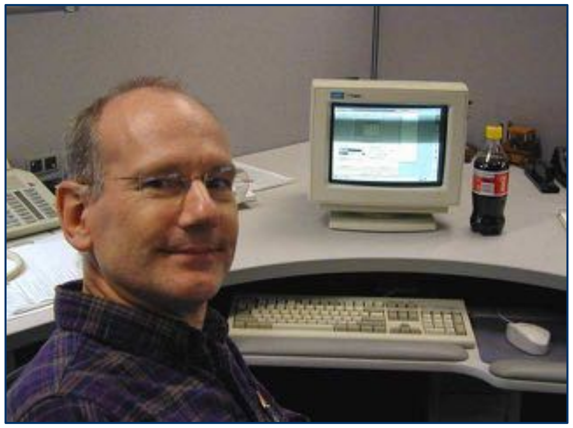
Mouse
Human
Cyto
SCF
Gene
SCF
Ori
Cyto
16C4
16C3.3
16C3.2
16C3.1
16C2
16C1.3
16C1.2
16C1.1
16B5
16B4
16B3
16B2
16B1
16A3
16A2
16A1



Chr 18
Chr 20

Acknowledgements

Art Delcher



Steven Salzberg

Mihai Pop

Stefan Kurtz

Mike Schatz