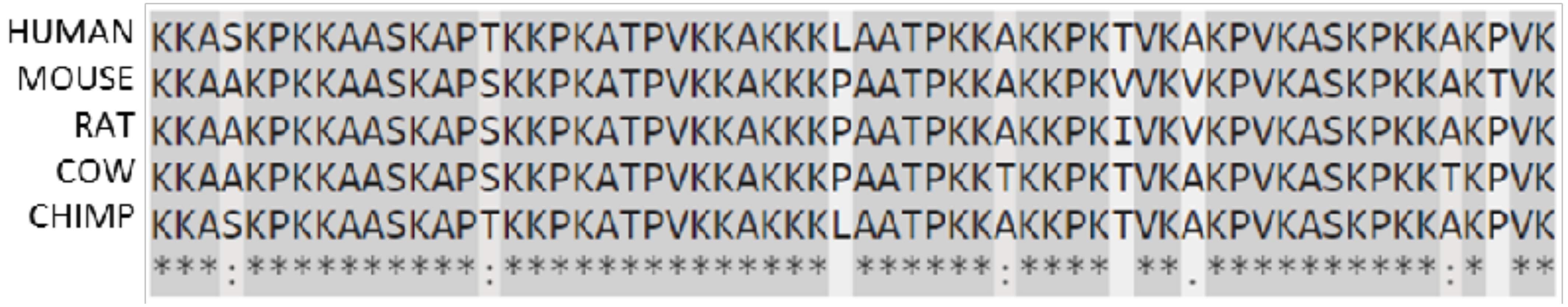
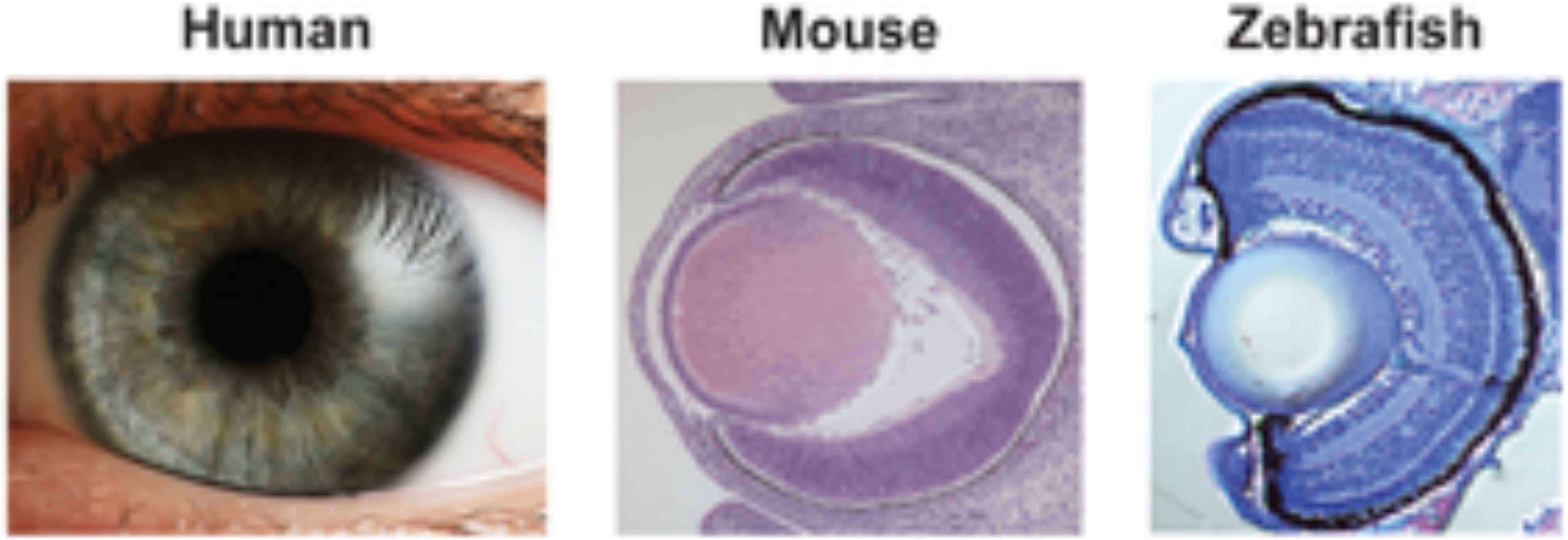
Identify Orthologies in Genomes

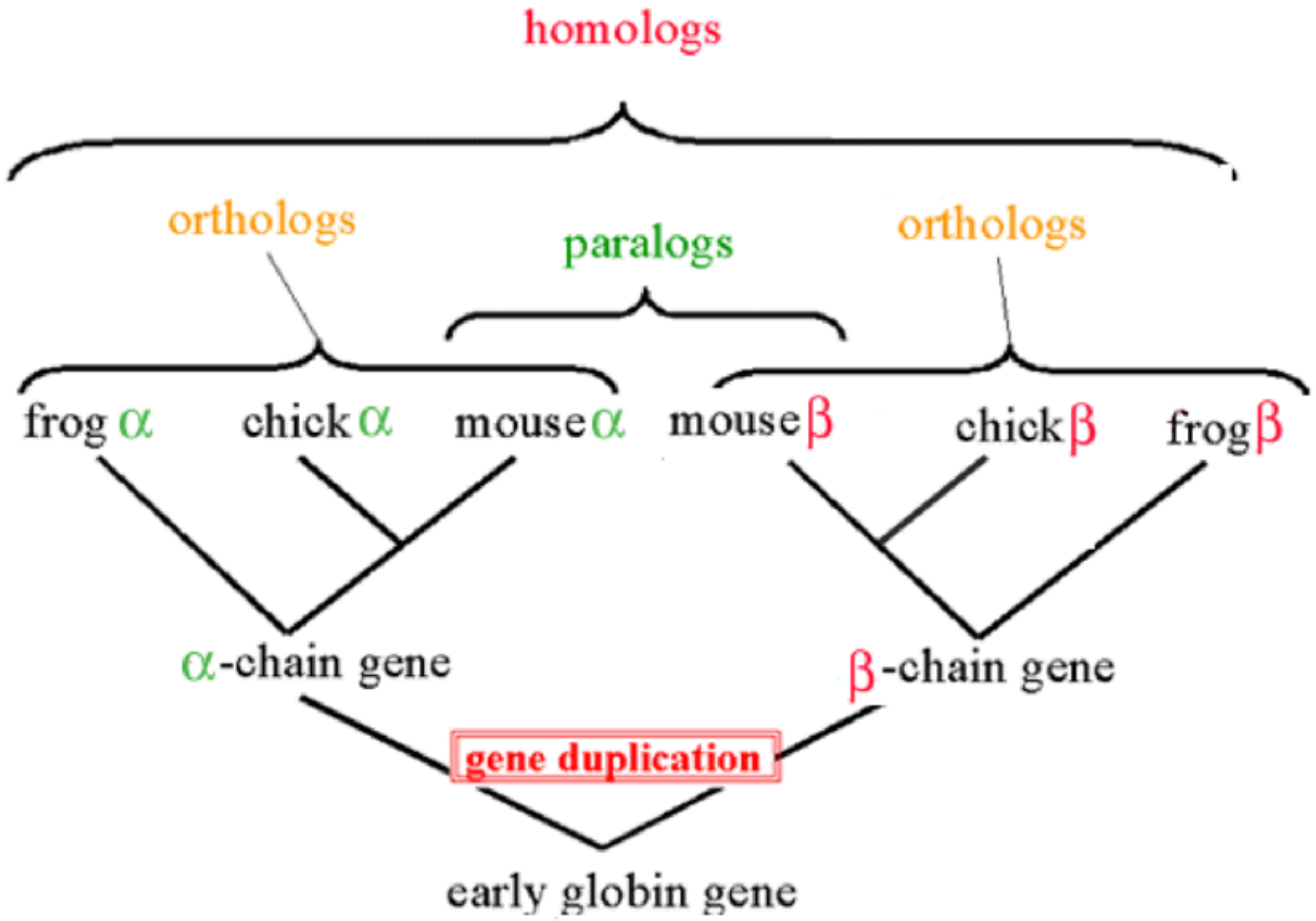
Histone H1 (residues 120-180)



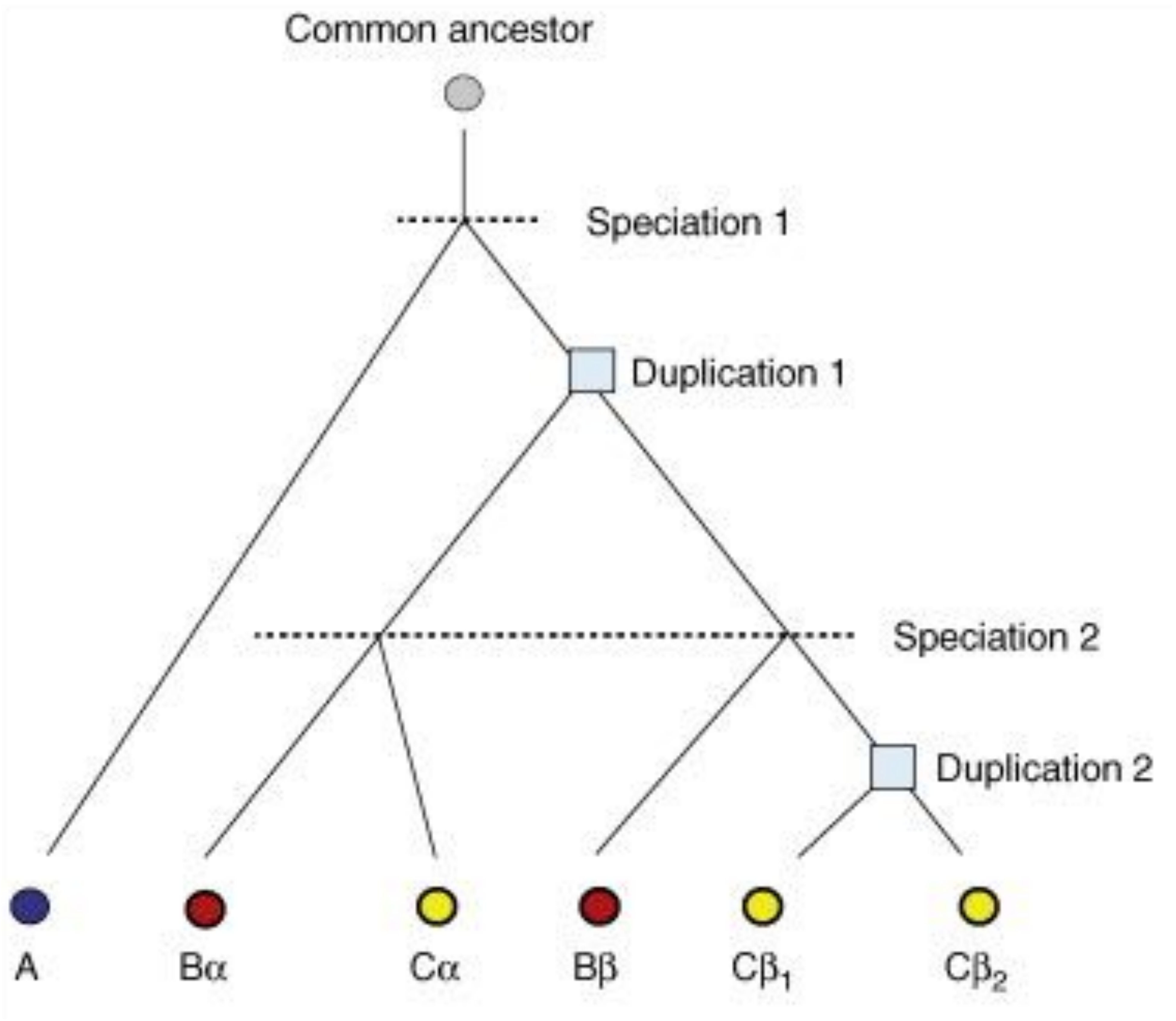
```
HUMAN  KKASKPKKAASKAPTKKPKATPVKKAKKKLAATPKKAKKPKTVKAKPVKASKPKKAKPVK
MOUSE  KKAAKPKKAASKAPSKKPKATPVKKAKKKPAATPKKAKKPKVVKVKPVKASKPKKAKTVK
  RAT  KKAAKPKKAASKAPSKKPKATPVKKAKKKPAATPKKAKKPKIVKVKPVKASKPKKAKPVK
  COW  KKAAKPKKAASKAPSKKPKATPVKKAKKKPAATPKKTKKPKTVKAKPVKASKPKKTKPVK
CHIMP  KKASKPKKAASKAPTKKPKATPVKKAKKKLAATPKKAKKPKTVKAKPVKASKPKKAKPVK
       ***:**********:************** ******:**** **.**********:* **
```


Human
Mouse
Zebrafish

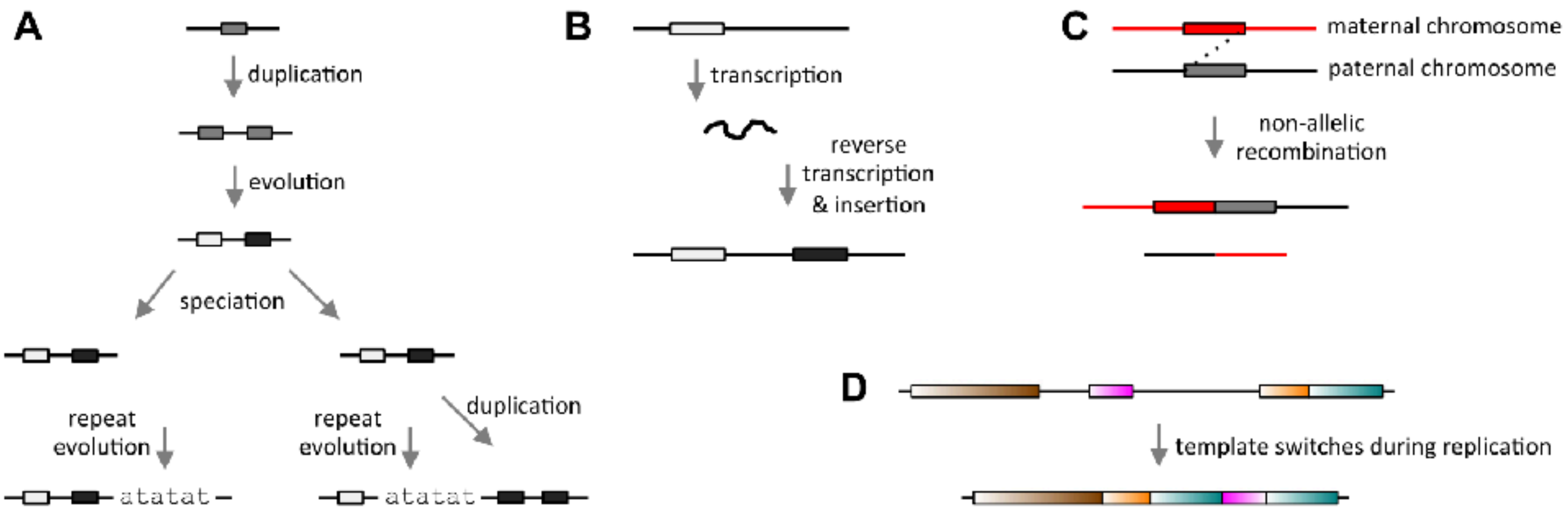
Orthologs and Paralogs



Some Facts About Orthologs

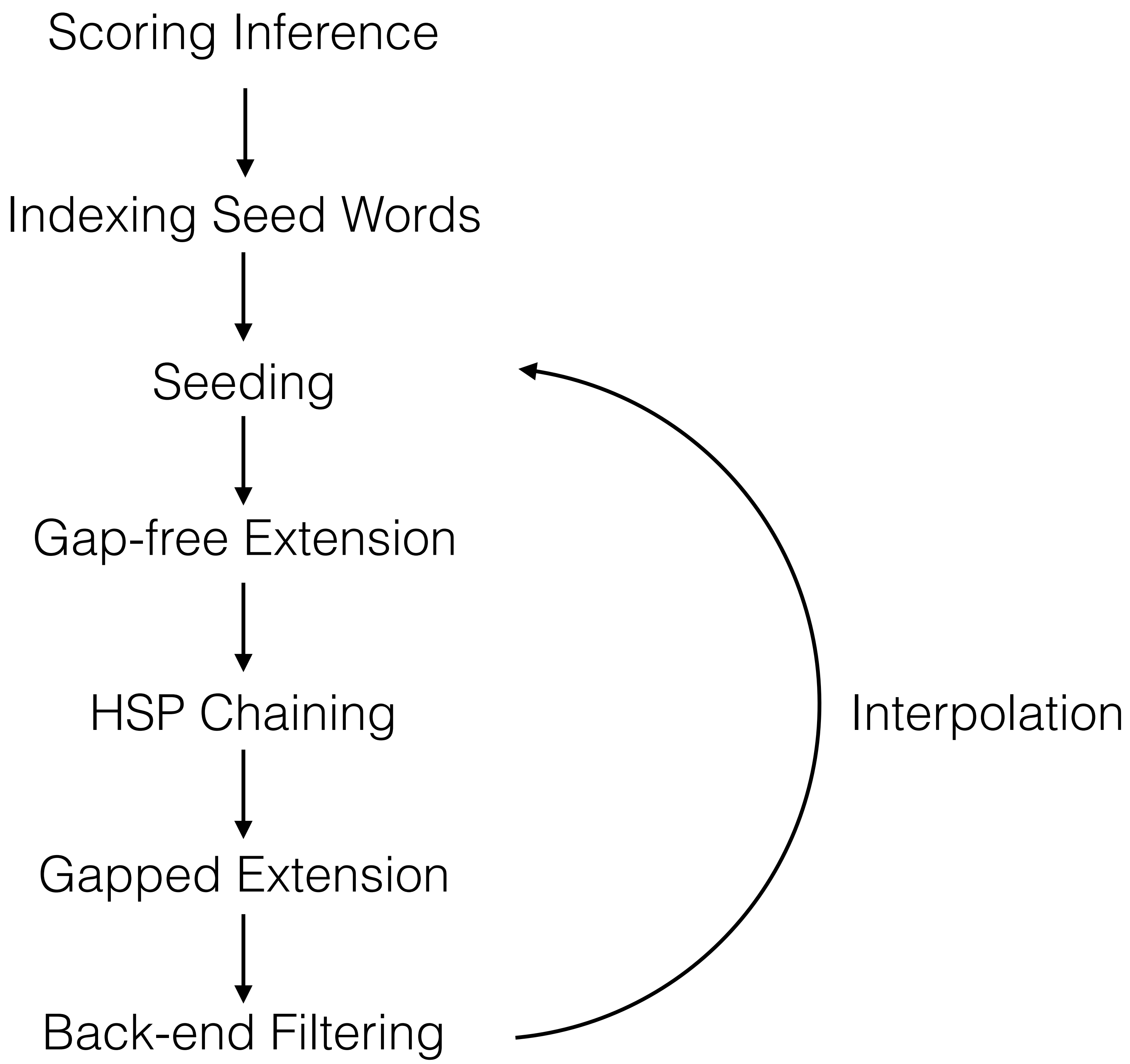


Some Facts About Orthologs

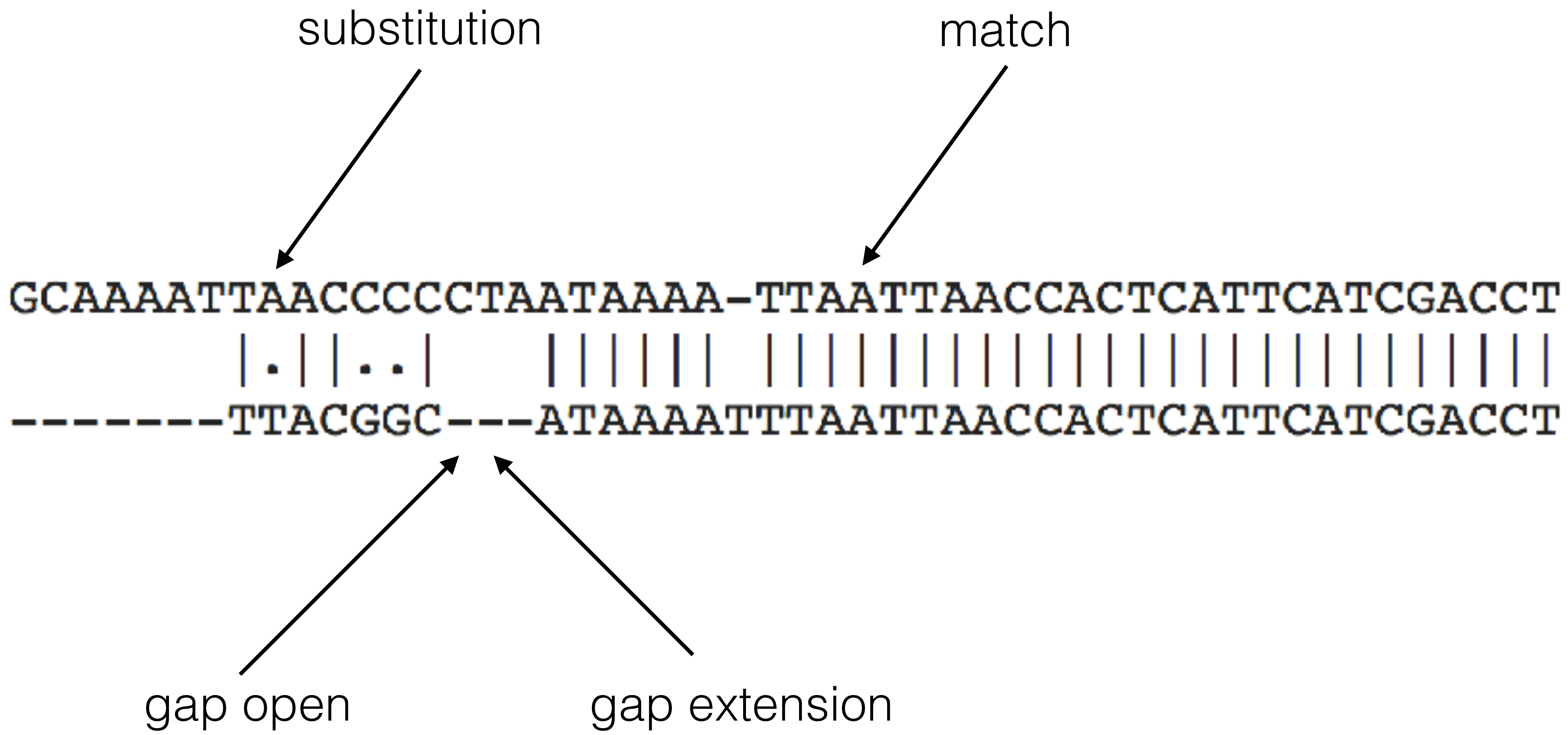


Pairwise Aligner — LASTZ

A Brief Overview



Scoring Inference


substitution
match
GCAAAATTAACCCCCTAATAAAA-TTAATTAACCACTCATTCATCGACCT
|.||..| |||||| |||||||||||||||||||||||||
-------TTACGGC---ATAAAATTTAATTAACCACTCATTCATCGACCT
gap open
gap extension

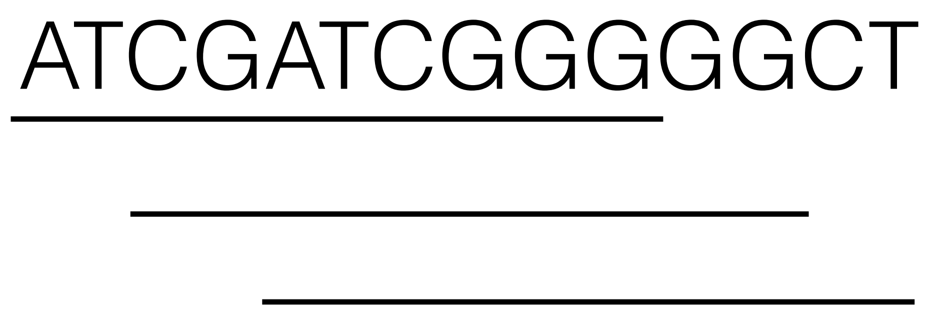
Infer the score by computing the probability of different alignment events estimated from alignments.

Optimization starts from a generic scoring set to create alignments.

Indexing Seed Words

k=10

step size=2



kmer	position
ATCGATCGGG	1
CGATCGGGGG	3
ATCGGGGGCT	5

Parses the target sequence(s) into overlapping seed words of some constant length, then word and position pairs are collected into a table, both for query and target

Seeding

Query Seed Table

kmer	position
ATCGATCGGG	1
CGATCGGGGG	3
ATCGGGGGCT	5

Target1 Seed Table

kmer	position
ATCGGGGGCT	8
CGGGGGCTTC	10
GGGGCTTCAA	12

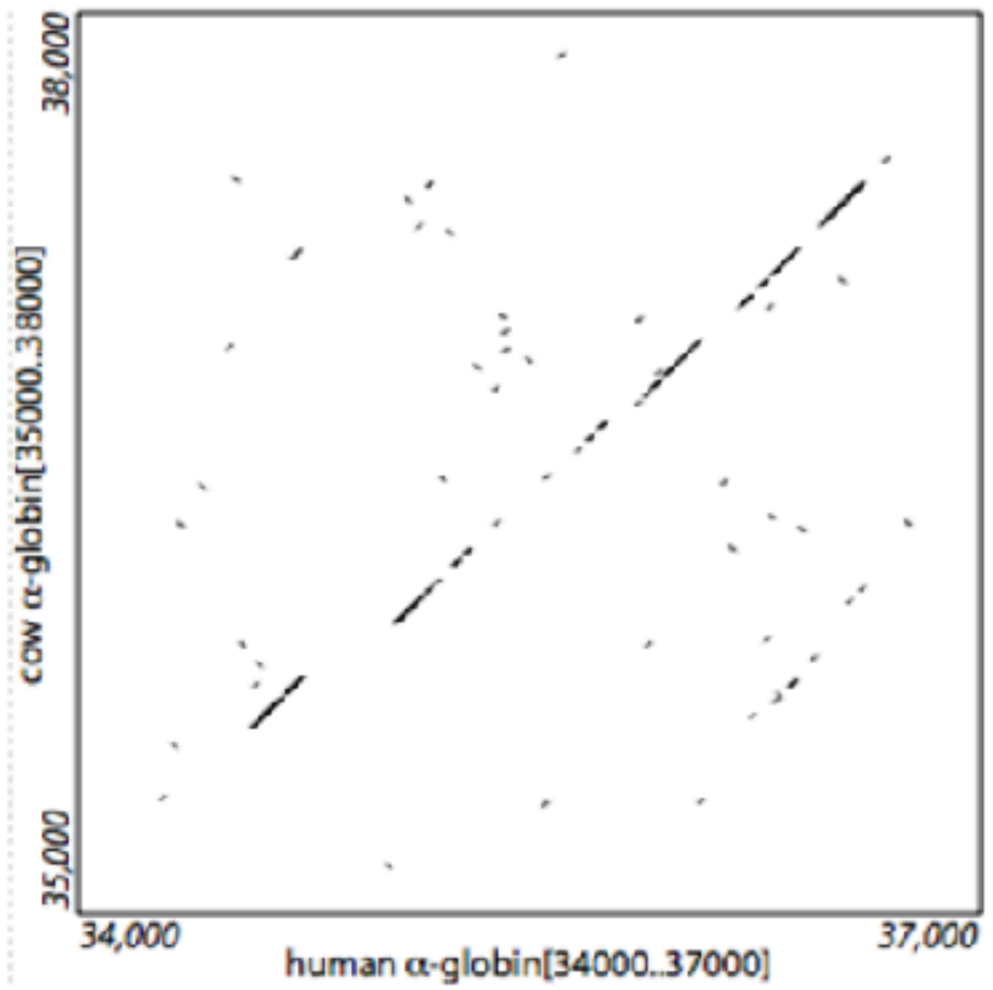
Target2 Seed Table

kmer	position
AT**AC**GGGGCT	8
CGGGGGCTTC	10
GGGGCTTCAA	12

Find seeds exact or near match
between target and query sequences

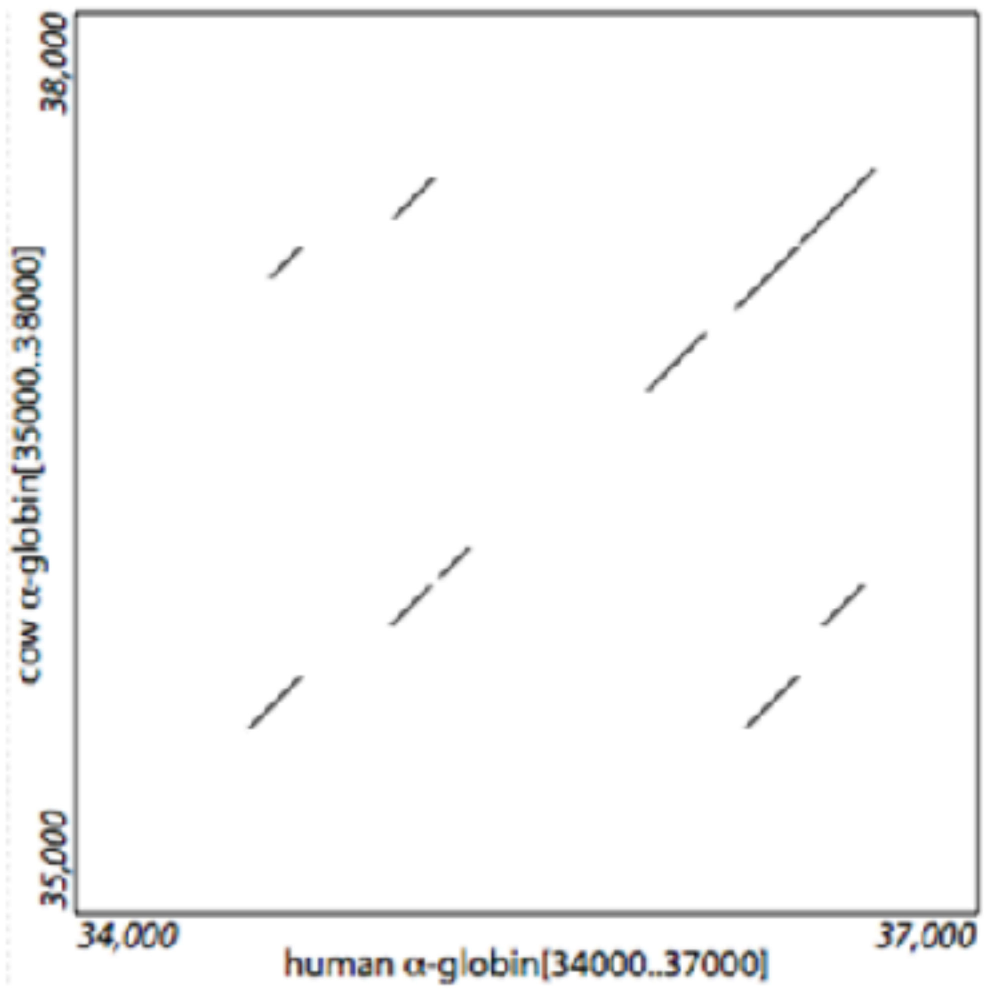
Gap-free Extension

Before Gap-free Extension



```
lastz \
  aglobin.2bit/human[34000..37000] \
  aglobin.2bit/cow[35000..38000] \
  --nogfextend --nochain --nogapped
```

After Gap-free Extension



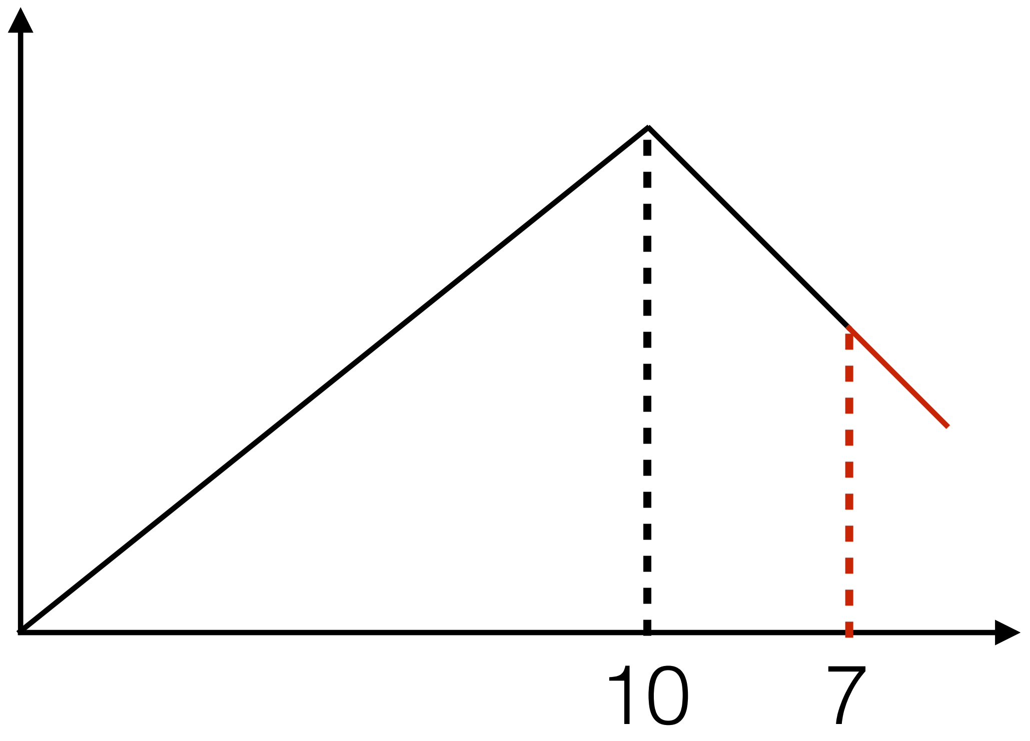
```
lastz \
  aglobin.2bit/human[34000..37000] \
  aglobin.2bit/cow[35000..38000] \
  --gfextend --nochain --nogapped
```

Each seed is extended along diagonal in both direction without allowing gaps to determine whether it is part of a high-scoring segment pair (HSP).

They extends following extension rules, currently either exact match, M-mismatch, or x-drop.

X-drop

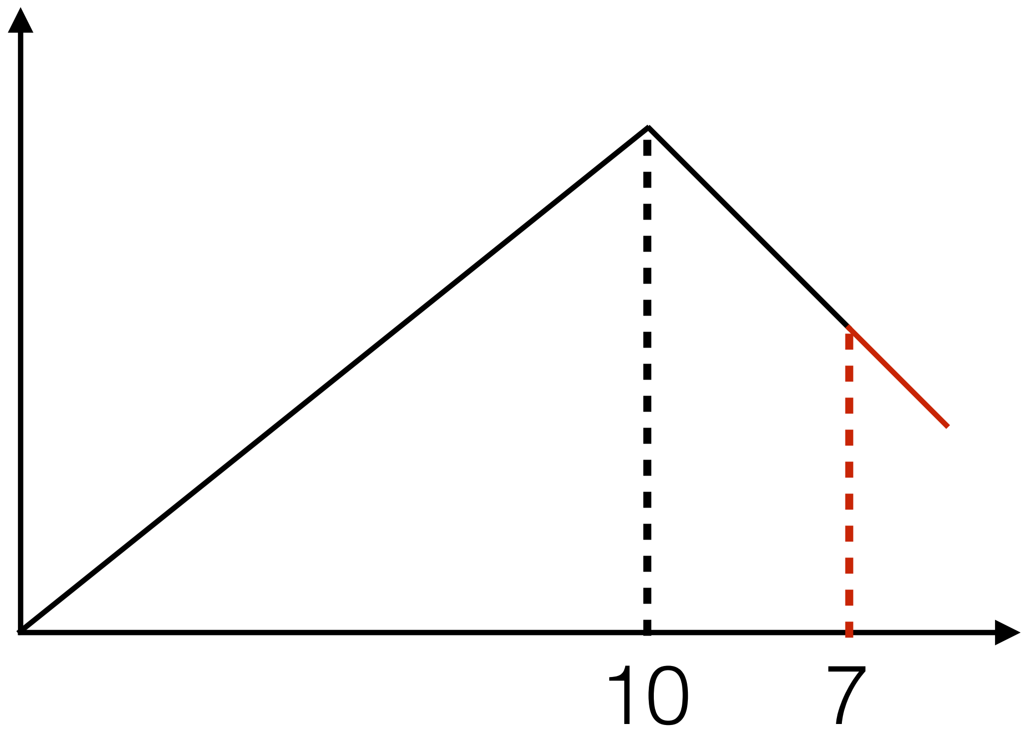
CAGCGGGCACATCGG
CAGCGGGCACTAGCC



match=1
mismatch=-1
x=3

X-drop

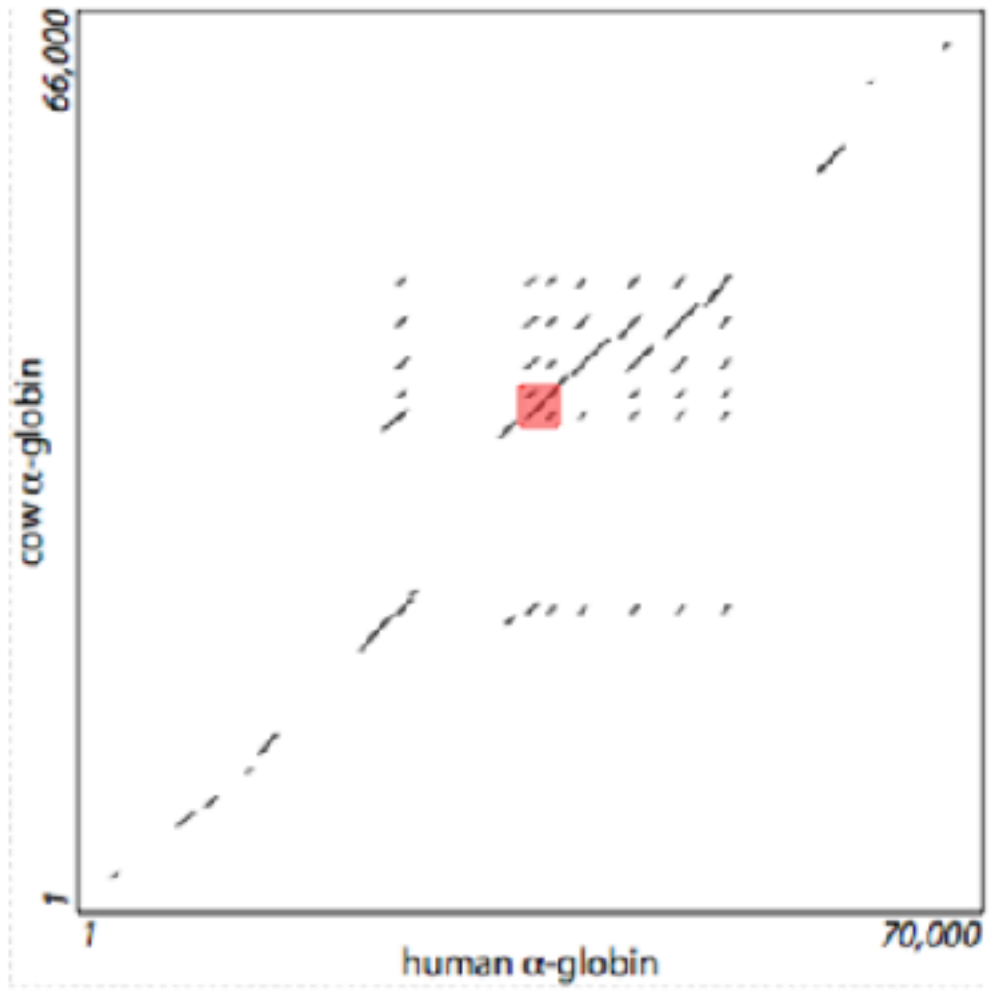
CAGCGGGCACATC
CAGCGGGCACTAG



match=1
mismatch=-1
x=3

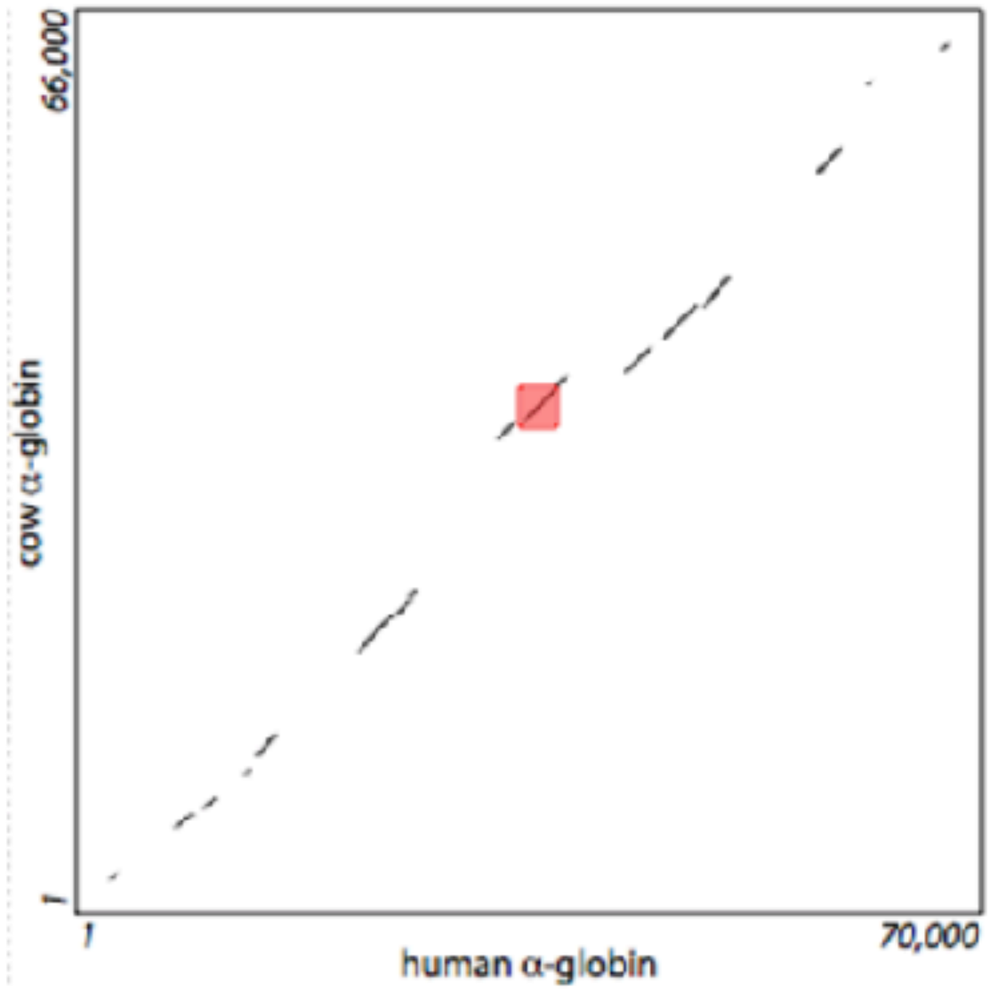
HSP Chaining

Before HSP Chaining



```
lastz \
  aglobin.2bit/human \
  aglobin.2bit/cow \
  --gfextend --nochain --gapped
```

After HSP Chaining

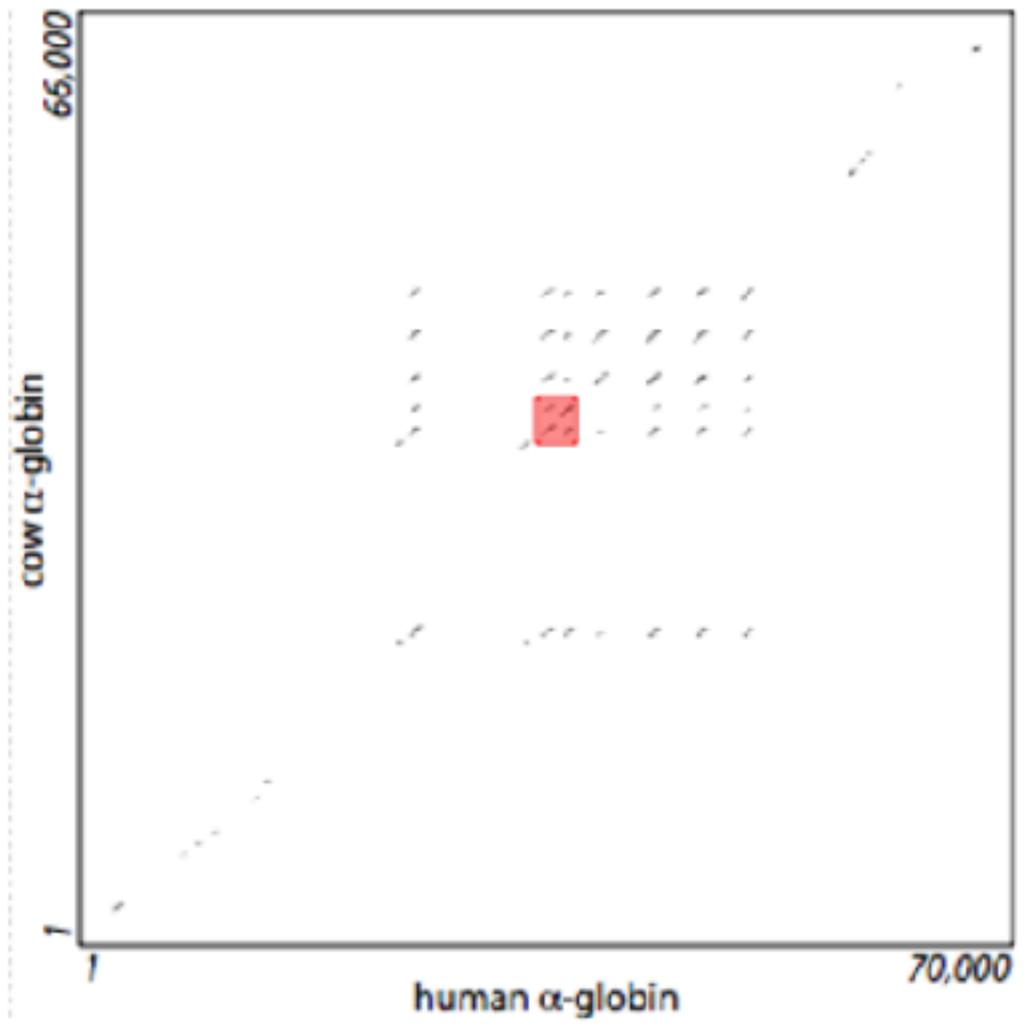


```
lastz \
  aglobin.2bit/human \
  aglobin.2bit/cow \
  --gfextend --chain --gapped
```

The chaining stage finds the highest scoring series of HSPs in which each HSP begins strictly before the start of the next. It's primary intend for HSPs in the same relative order and orientation in the query as in the target

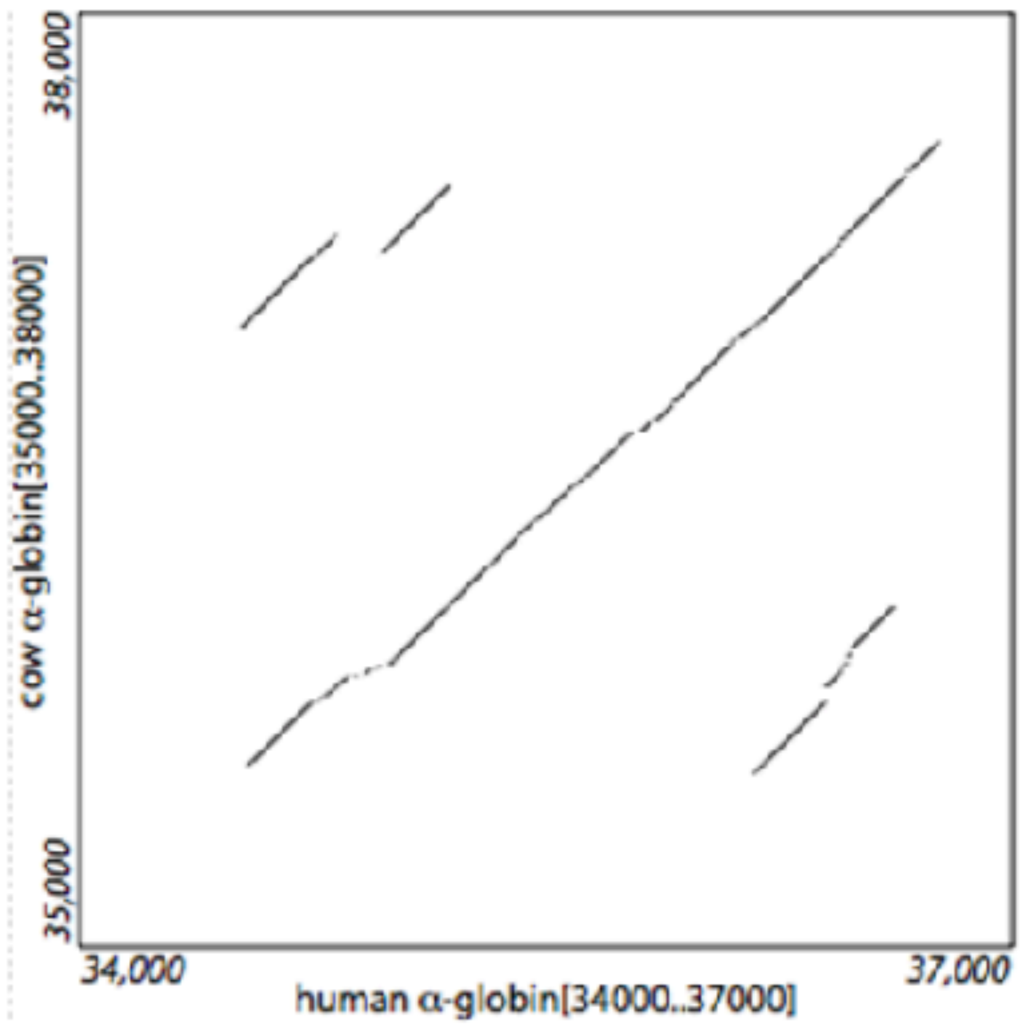
Gapped Extension

Before Gapped Extension



```
lastz \
  aglobin.2bit/human \
  aglobin.2bit/cow \
  --gfextend --nochain --nogapped
```

After Gapped Extension



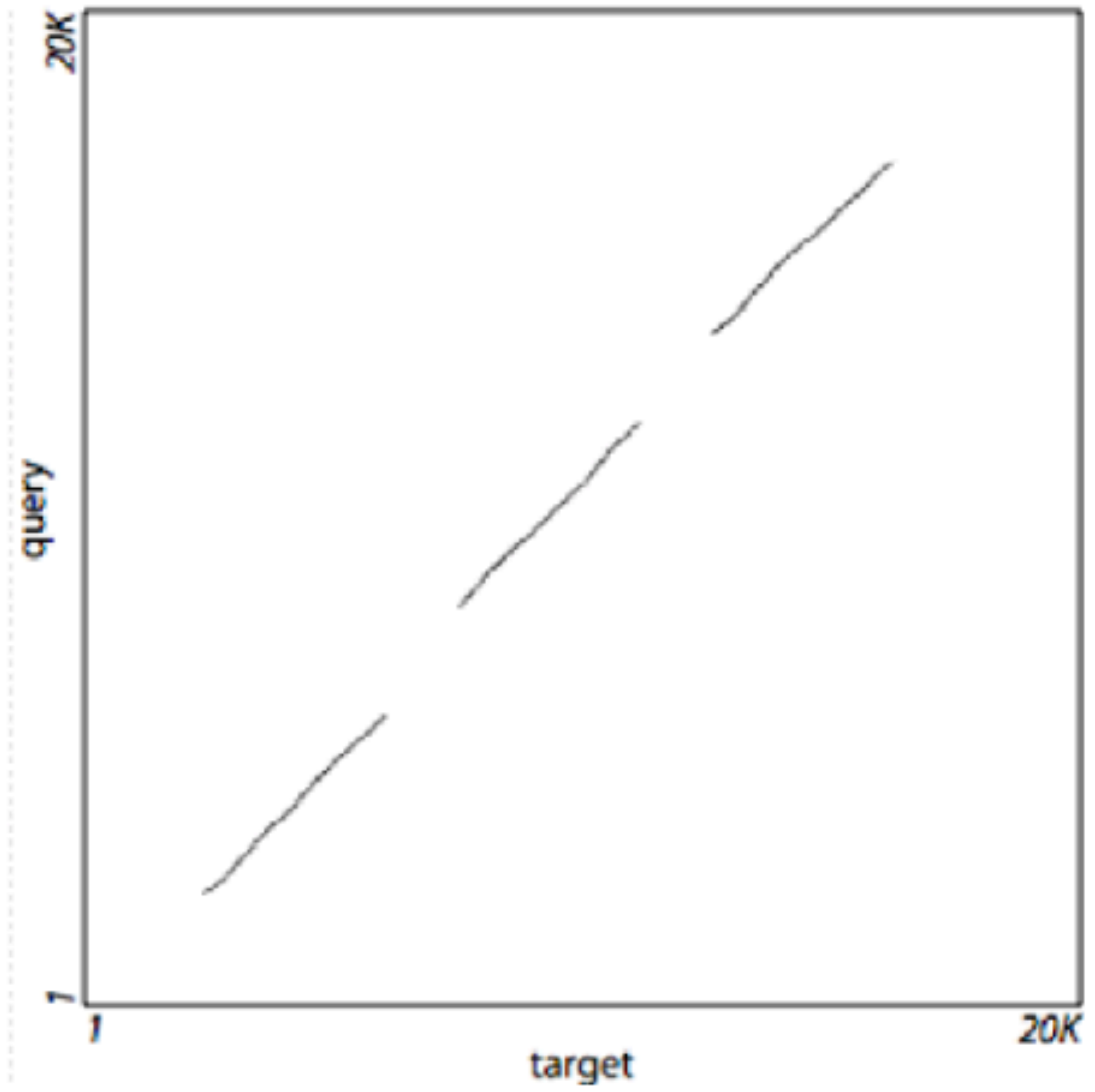
```
lastz \
  aglobin.2bit/human[34000..37000] \
  aglobin.2bit/cow[35000..38000] \
  --gfextend --nochain --gapped
```

Each HSP is first reduced to a single anchor point, then gapped extension is performed independently in both directions from the anchor point

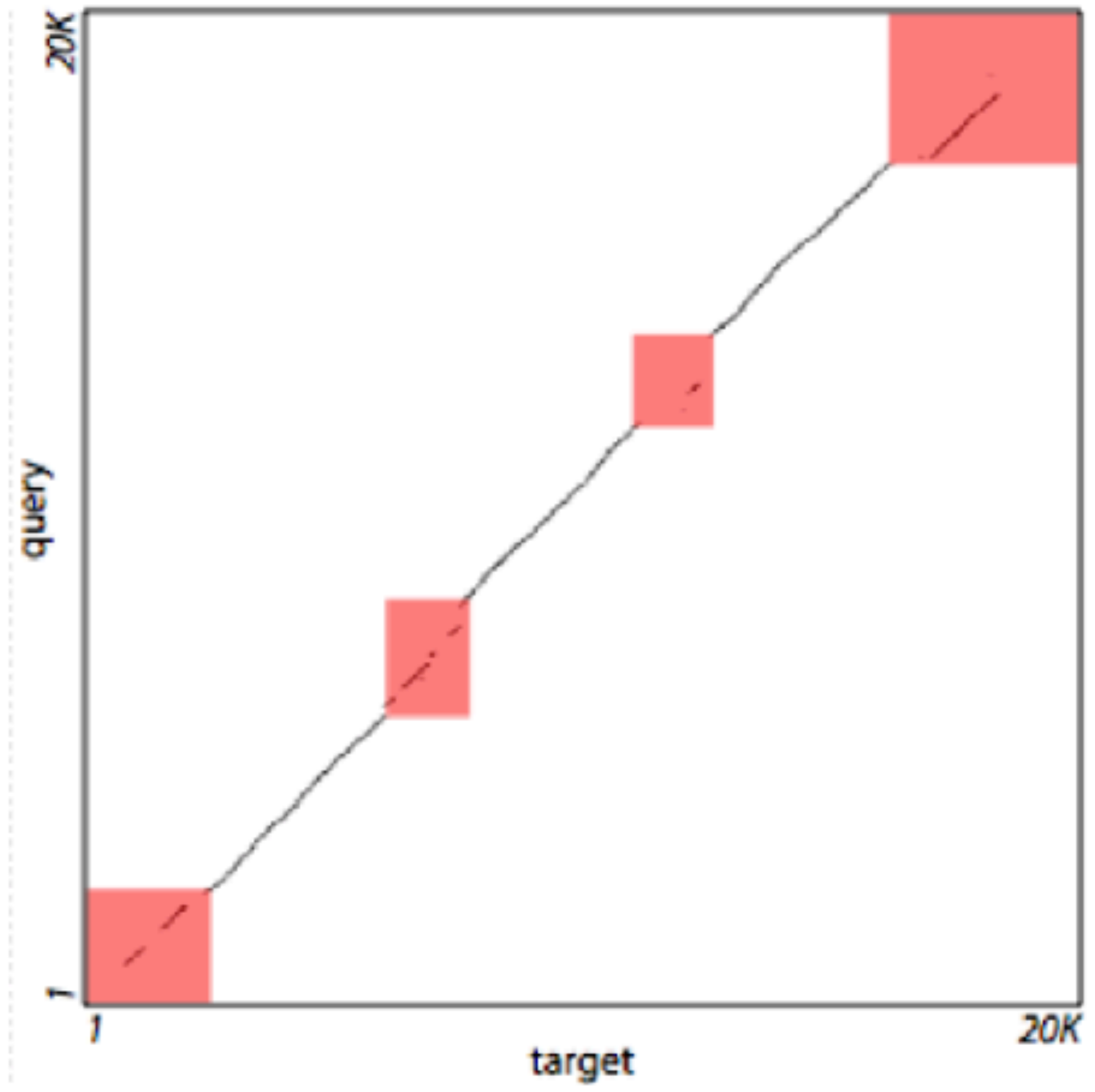
Back-end Filtering

Whatever alignment blocks have made it through the above gauntlet are then subjected to identity, continuity, coverage and match count filtering. Blocks that do not meet the specified range for each feature are discarded.

Interpolation



`lastz target query`



`lastz target query --inner=1000`

Using high sensitivity to run another complete alignment round (seeding, gap-free extension, chaining, gapped extension and back-end filtering)

BLAST vs LASTZ

	Discontiguous Mega BLAST	LASTZ
Remove Low Complexity and Repeats	DUST	Repeat Masker
Scoring Inference	Fixed Score	Iterative Inference/HOXD70
Seeding	Spaced Seeds	Spaced Seeds
Gap-free Extension	—	x-drop
Chaining	—	✓
Gapped Extension	y-drop	y-drop
Evaluation	E-value	Back-end Filtering
Interpolation	—	✓

Implementation Time

Platform, Compiler and Other Software Required

LASTZ supports Macintosh OS X, Unix and Linux platforms

LASTZ is written in C and compiled with gcc (4.8.5 on my Macbook)

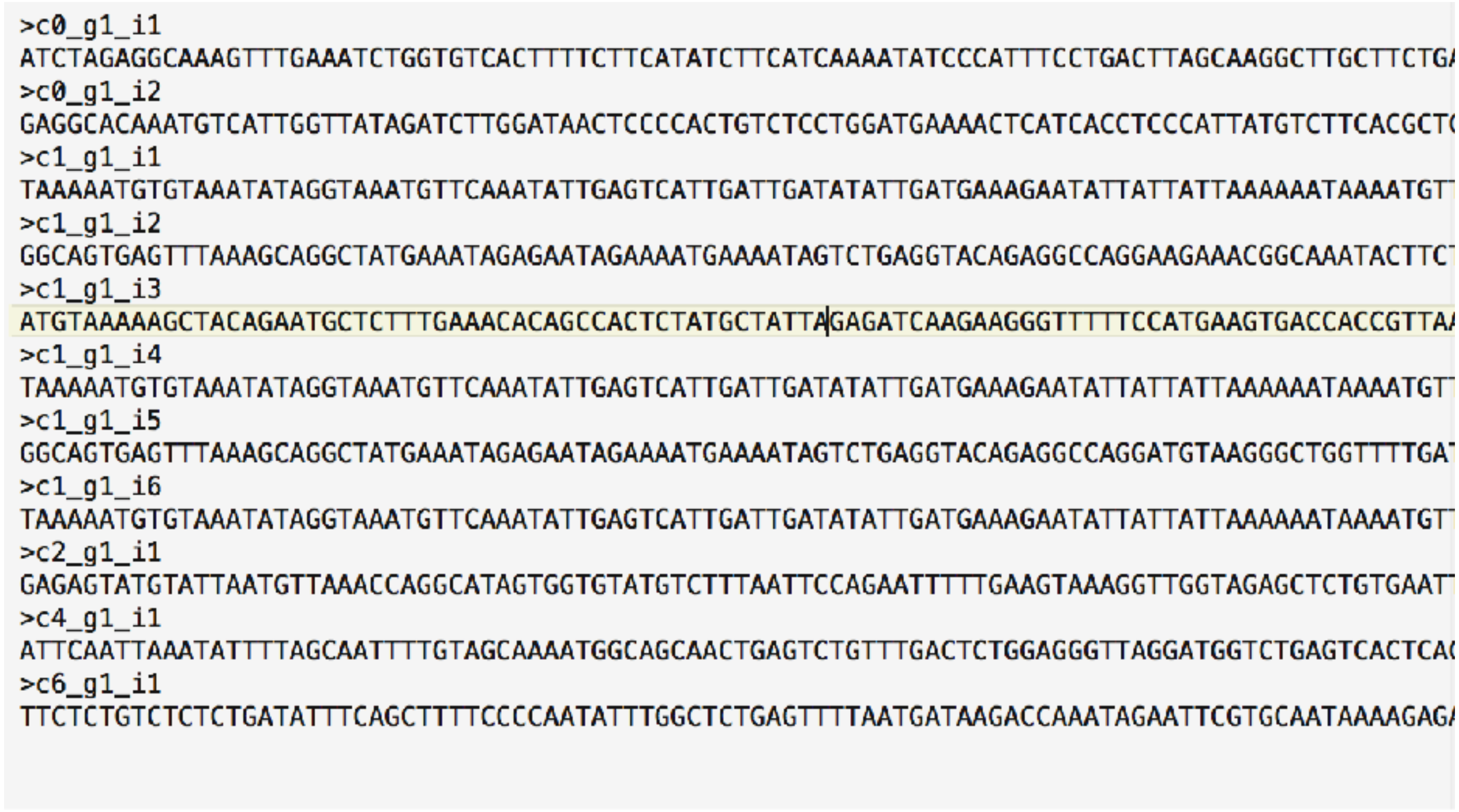
Image of Dotplot output can be generated by R and R studio

Installation

```
tar -zxvf lastz-1.02.00.tar.gz

cd <somepath>/lastz-1.02.00.tar.gz/src

make

make install
```

Add lastz to $PATH

Common Input Files

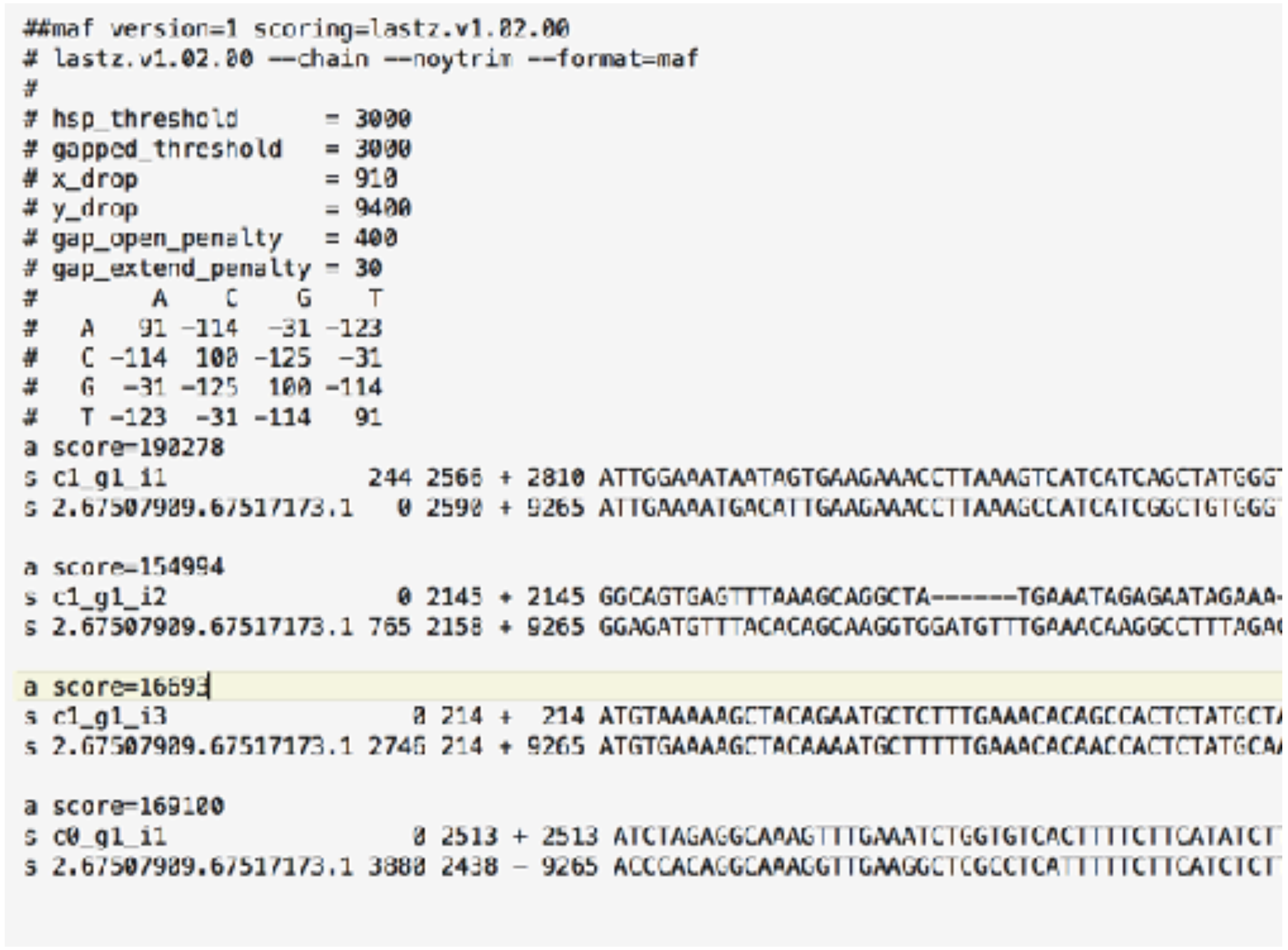


```
>c0_g1_i1
ATCTAGAGGCAAAGTTTGAAATCTGGTGTCACTTTTCTTCATATCTTCATCAAAATATCCCATTTCCTGACTTAGCAAGGCTTGCTTCTG
>c0_g1_i2
GAGGCACAAATGTCATTGGTTATAGATCTTGGATAACTCCCCACTGTCTCCTGGATGAAAACTCATCACCTCCCATTATGTCTTCACGCT
>c1_g1_i1
TAAAAATGTGTAAATATAGGTAAATGTTCAAATATTGAGTCATTGATTGATATATTGATGAAAGAATATTATTATTAAAAAATAAAATGT
>c1_g1_i2
GGCAGTGAGTTTAAAGCAGGCTATGAAATAGAGAATAGAAAATGAAAATAGTCTGAGGTACAGAGGCCAGGAAGAAACGGCAAATACTTC
>c1_g1_i3
ATGTAAAAAGCTACAGAATGCTCTTTGAAACACAGCCACTCTATGCTATTAGAGATCAAGAAGGGTTTTTCCATGAAGTGACCACCGTTA
>c1_g1_i4
TAAAAATGTGTAAATATAGGTAAATGTTCAAATATTGAGTCATTGATTGATATATTGATGAAAGAATATTATTATTAAAAAATAAAATGT
>c1_g1_i5
GGCAGTGAGTTTAAAGCAGGCTATGAAATAGAGAATAGAAAATGAAAATAGTCTGAGGTACAGAGGCCAGGATGTAAGGGCTGGTTTTGA
>c1_g1_i6
TAAAAATGTGTAAATATAGGTAAATGTTCAAATATTGAGTCATTGATTGATATATTGATGAAAGAATATTATTATTAAAAAATAAAATGT
>c2_g1_i1
GAGAGTATGTATTAATGTTAAACCAGGCATAGTGGTGTATGTCTTTAATTCCAGAATTTTTGAAGTAAAGGTTGGTAGAGCTCTGTGAAT
>c4_g1_i1
ATTCAATTAAATATTTTAGCAATTTTGTAGCAAAATGGCAGCAACTGAGTCTGTTTGACTCTGGAGGGTTAGGATGGTCTGAGTCACTCA
>c6_g1_i1
TTCTCTGTCTCTCTGATATTTCAGCTTTTCCCCAATATTTGGCTCTGAGTTTTAATGATAAGACCAAATAGAATTCGTGCAATAAAAGAG
```

Query and target files in Fasta format

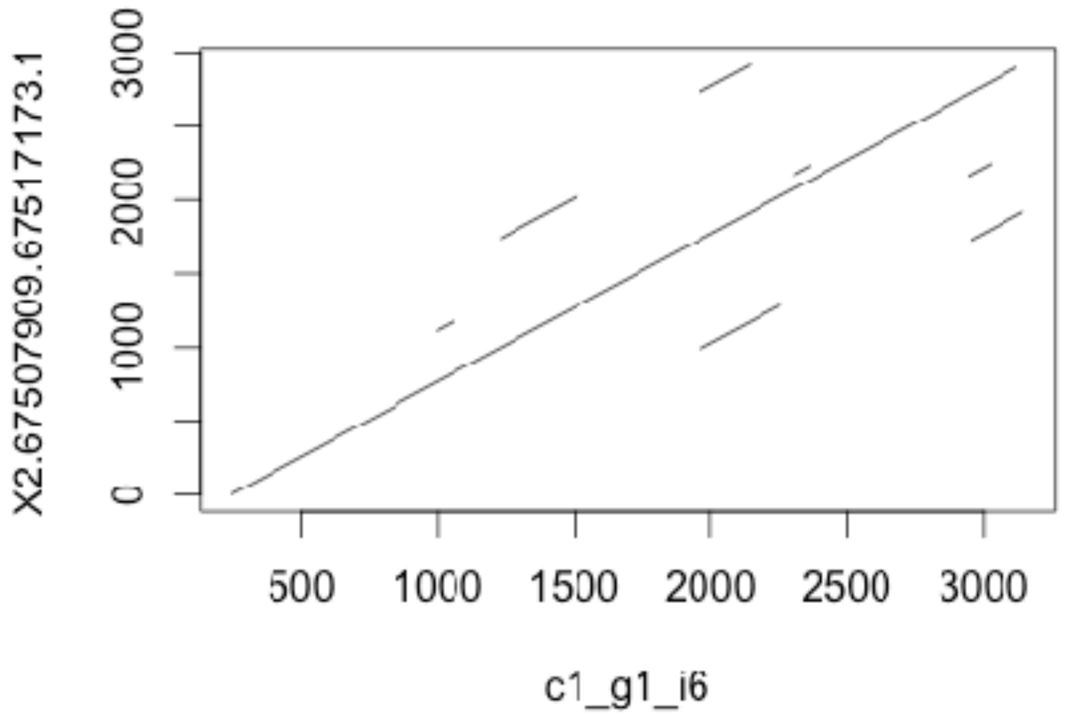
Common Output Files Format

Maf Format Output



```
##maf version=1 scoring=lastz.v1.02.00
# lastz.v1.02.00 --chain --noytrim --format=maf
#
# hsp_threshold      = 3000
# gapped_threshold   = 3000
# x_drop             = 910
# y_drop             = 9400
# gap_open_penalty   = 400
# gap_extend_penalty = 30
#         A    C    G    T
#   A    91 -114  -31 -123
#   C  -114  100 -125  -31
#   G   -31 -125  100 -114
#   T  -123  -31 -114   91
a score=190278
s c1_g1_i1                244 2566 + 2810 ATTGGAAATAATAGTGAAGAAACCTTAAAGTCATCATCAGCTATGGGT
s 2.67507989.67517173.1     0 2590 + 9265 ATTGAAAATGACATTGAAGAAACCTTAAAGCCATCATCGGCTGTGGGT

a score=154994
s c1_g1_i2                  0 2145 + 2145 GGCAGTGAGTTTAAAGCAGGCTA-------TGAAATAGAGAATAGAAA-
s 2.67507989.67517173.1   765 2158 + 9265 GGAGATGTTTACACAGCAAGGTGGATGTTTGAAACAAGGCCTTTAGAC

a score=16693
s c1_g1_i3                  0  214 +  214 ATGTAAAAAGCTACAGAATGCTCTTTGAAACACAGCCACTCTATGCTA
s 2.67507989.67517173.1  2746  214 + 9265 ATGTGAAAAGCTACAAAATGCTTTTTGAAACACAACCACTCTATGCAA

a score=169100
s c0_g1_i1                  0 2513 + 2513 ATCTAGAGGCAAAGTTTGAAATCTGGTGTCACTTTTCTTCATATCT
s 2.67507989.67517173.1  3880 2438 - 9265 ACCCACAGGCAAAGGTTGAAGGCTCGCCTCATTTTTCTTCATCTCT
```

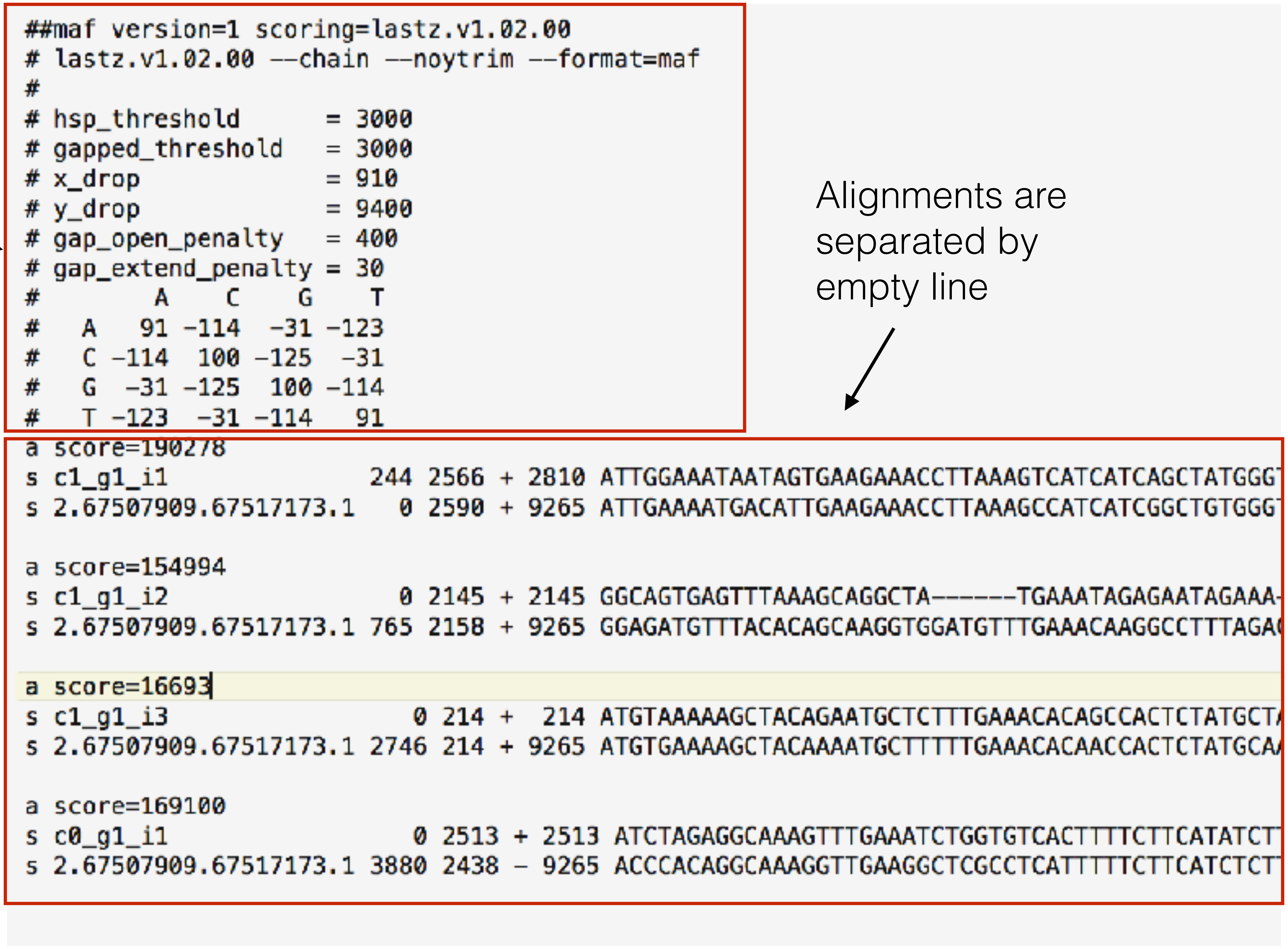
R Dotplot



Maf Format

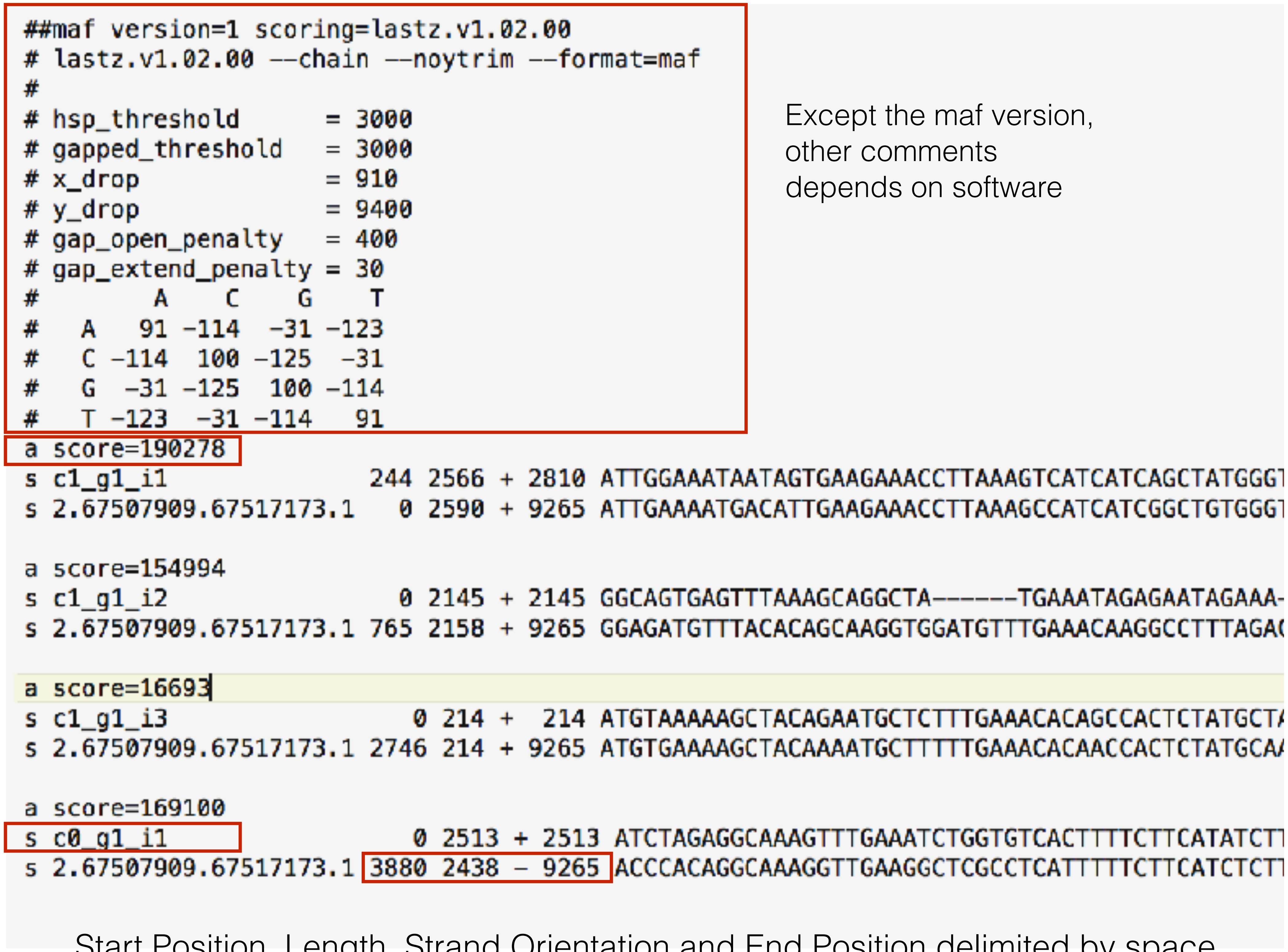
Comments

Alignments are separated by empty line



```
##maf version=1 scoring=lastz.v1.02.00
# lastz.v1.02.00 --chain --noytrim --format=maf
#
# hsp_threshold      = 3000
# gapped_threshold   = 3000
# x_drop             = 910
# y_drop             = 9400
# gap_open_penalty   = 400
# gap_extend_penalty = 30
#        A    C    G    T
#   A   91 -114  -31 -123
#   C -114  100 -125  -31
#   G  -31 -125  100 -114
#   T -123  -31 -114   91
a score=190278
s c1_g1_i1                 244 2566 + 2810 ATTGGAAATAATAGTGAAGAAACCTTAAAGTCATCATCAGCTATGGG
s 2.67507909.67517173.1      0 2590 + 9265 ATTGAAAATGACATTGAAGAAACCTTAAAGCCATCATCGGCTGTGGG

a score=154994
s c1_g1_i2                   0 2145 + 2145 GGCAGTGAGTTTAAAGCAGGCTA------TGAAATAGAGAATAGAAA
s 2.67507909.67517173.1    765 2158 + 9265 GGAGATGTTTACACAGCAAGGTGGATGTTTGAAACAAGGCCTTTAGA

a score=16693
s c1_g1_i3                    0 214 +  214 ATGTAAAAAGCTACAGAATGCTCTTTGAAACACAGCCACTCTATGCT
s 2.67507909.67517173.1    2746 214 + 9265 ATGTGAAAAGCTACAAAATGCTTTTTGAAACACAACCACTCTATGCA

a score=169100
s c0_g1_i1                    0 2513 + 2513 ATCTAGAGGCAAAGTTTGAAATCTGGTGTCACTTTTCTTCATATCT
s 2.67507909.67517173.1    3880 2438 - 9265 ACCCACAGGCAAAGGTTGAAGGCTCGCCTCATTTTTCTTCATCTCT
```

Maf Format



```
##maf version=1 scoring=lastz.v1.02.00
# lastz.v1.02.00 --chain --noytrim --format=maf
#
# hsp_threshold      = 3000
# gapped_threshold   = 3000
# x_drop             = 910
# y_drop             = 9400
# gap_open_penalty   = 400
# gap_extend_penalty = 30
#        A    C    G    T
#   A   91 -114  -31 -123
#   C -114  100 -125  -31
#   G  -31 -125  100 -114
#   T -123  -31 -114   91
a score=190278
s c1_g1_i1                244 2566 + 2810 ATTGGAAATAATAGTGAAGAAACCTTAAAGTCATCATCAGCTATGGGT
s 2.67507909.67517173.1     0 2590 + 9265 ATTGAAAATGACATTGAAGAAACCTTAAAGCCATCATCGGCTGTGGGT

a score=154994
s c1_g1_i2                  0 2145 + 2145 GGCAGTGAGTTTAAAGCAGGCTA------TGAAATAGAGAATAGAAA-
s 2.67507909.67517173.1   765 2158 + 9265 GGAGATGTTTACACAGCAAGGTGGATGTTTGAAACAAGGCCTTTAGAC

a score=16693
s c1_g1_i3                  0 214 +  214 ATGTAAAAAGCTACAGAATGCTCTTTGAAACACAGCCACTCTATGCTA
s 2.67507909.67517173.1  2746 214 + 9265 ATGTGAAAAGCTACAAAATGCTTTTTGAAACACAACCACTCTATGCAA

a score=169100
s c0_g1_i1                  0 2513 + 2513 ATCTAGAGGCAAAGTTTGAAATCTGGTGTCACTTTTCTTCATATCTT
s 2.67507909.67517173.1  3880 2438 - 9265 ACCCACAGGCAAAGGTTGAAGGCTCGCCTCATTTTTCTTCATCTCTT
```

Except the maf version, other comments depends on software

“a” indicates the start of the block followed by alignment score

“s” indicates the start of the sequence followed by sequence ID

Start Position, Length, Strand Orientation and End Position delimited by space

Command Line Syntax

```
lastz <target> [<query>] [<options>]
```

```
lastz target.fas[multiple] query.fas  \
        --format=maf > alignment.maf
```

Try LASTZ on Gene Enrichment Data

Multiple Sequences Aligner — Mauve & ProgressiveMauve

A Brief Overview of Mauve

Find local alignments (multi-MUMs).

↓

Use the multi-MUMs to calculate a phylogenetic guide tree

↓

Select a subset of the multi-MUM to use as anchors, then partitioned into collinear groups call LCBs

↓

Perform recursive anchoring to identify additional alignment anchors within and outside each LCBs

↓

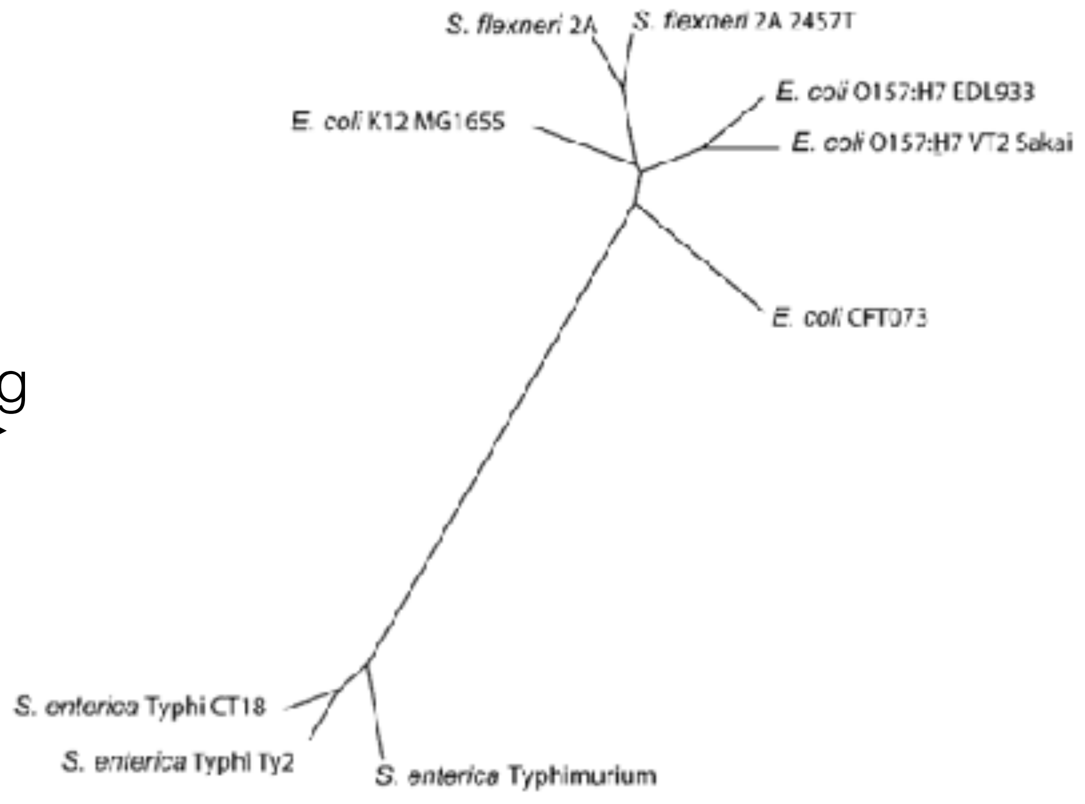
Perform a progressive alignment of each LCB using the guide tree

Find local alignments (multi-MUMs)

$$\text{Seeding (Exact match seed)} \longrightarrow \text{Gap-free Extension}$$

Calculating a Guide Tree

Gap-free multi-MUMs coverage $\xrightarrow{\text{Neighbor Joining}}$



Guide Tree

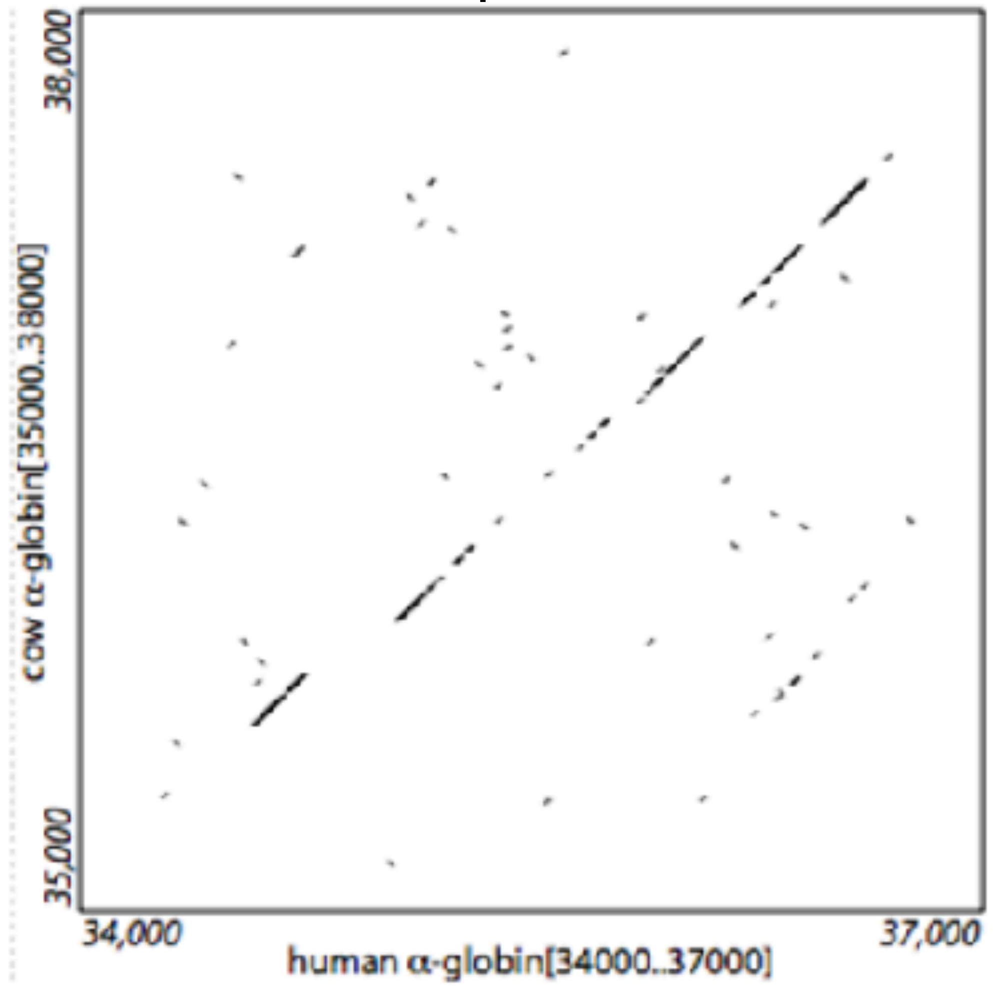
Partition Subset of the Multi-MUM into Locally Collinear Blocks LCBs

Delete spurious matches according to weight (length)

Determine a ear blocks

Calc ck

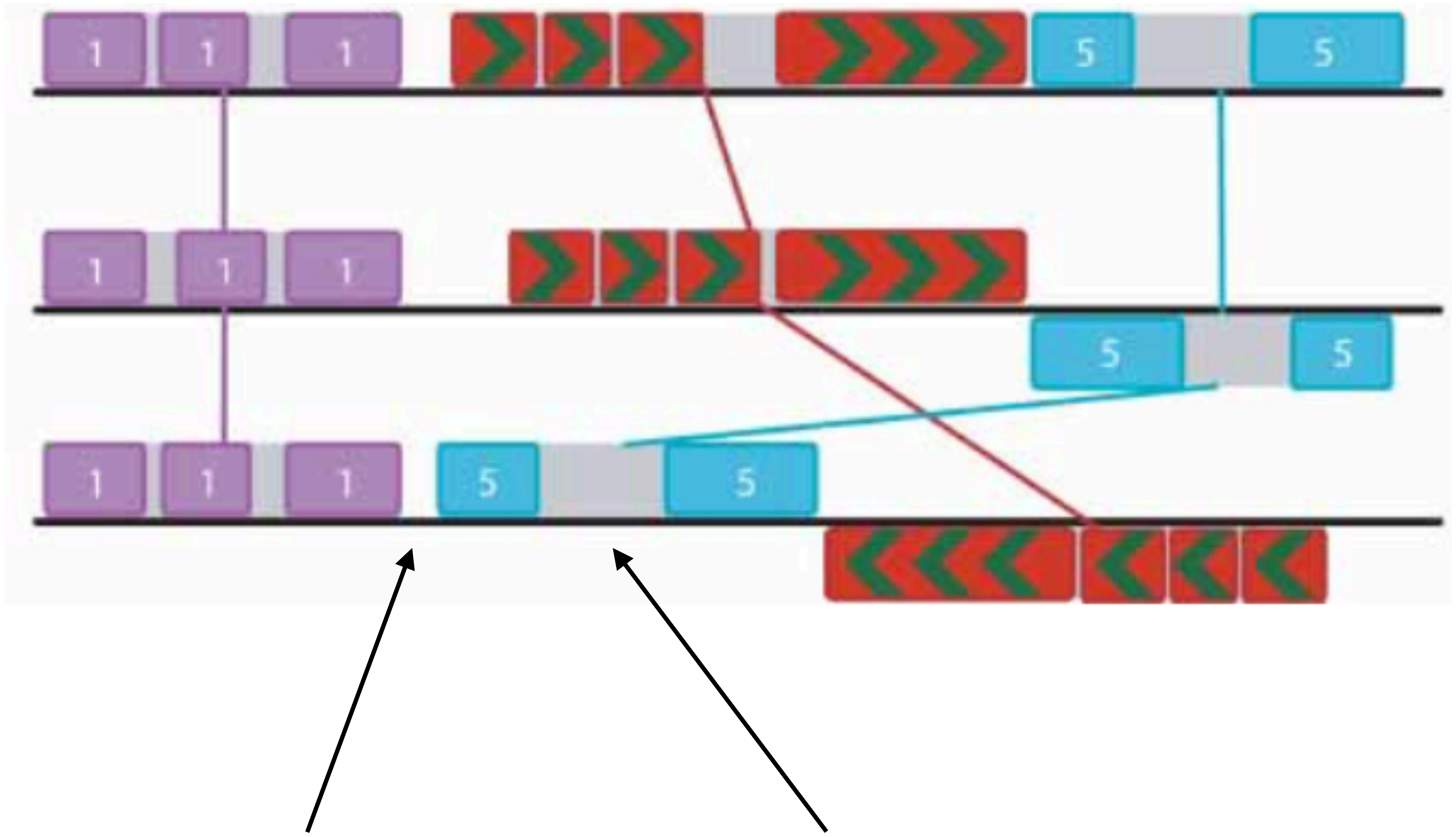
Stop if colli the Weight greater tha



Identify the collinear subsets which minimum weight is equal or greater than threshold

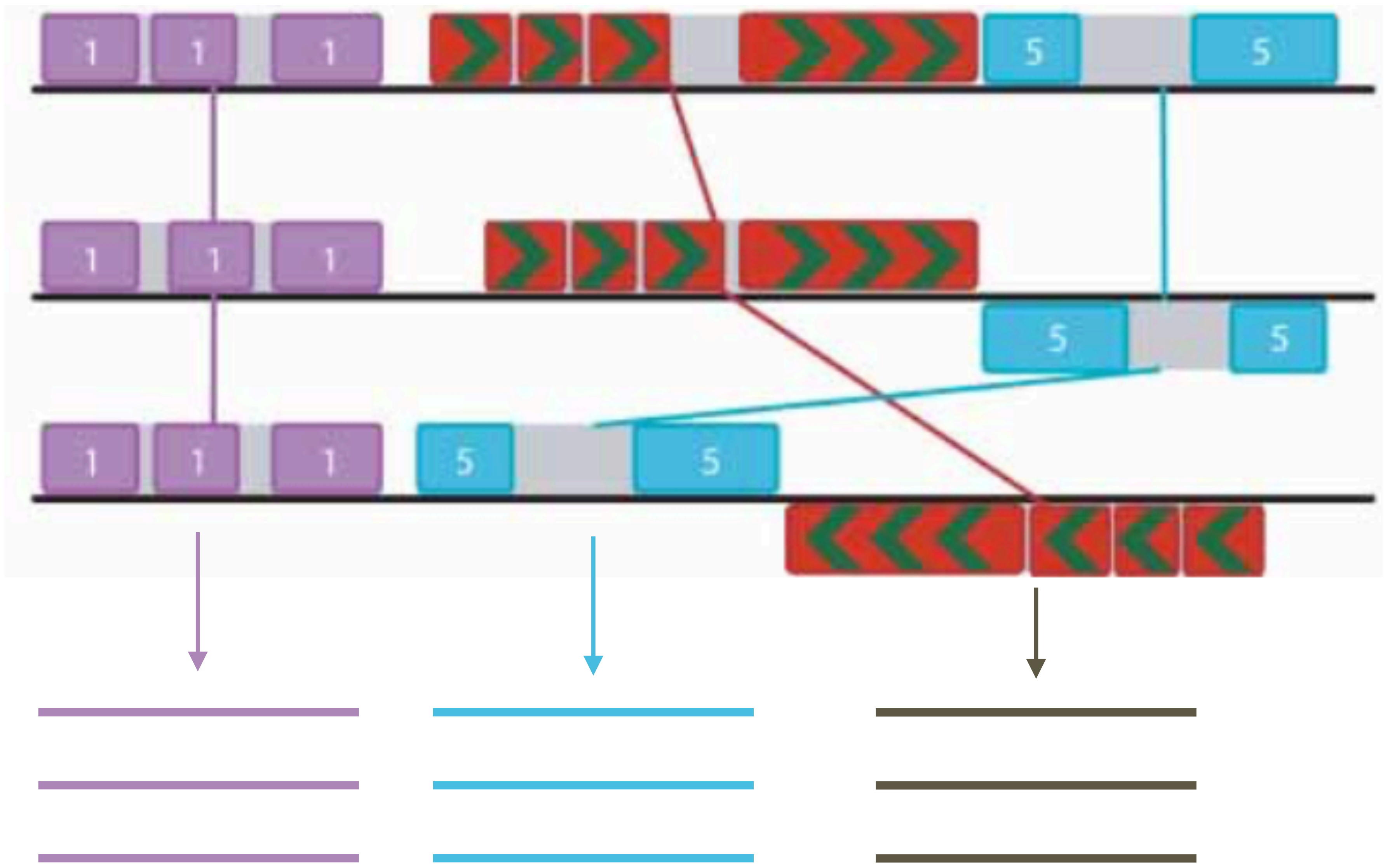
Remove the Multi-MUMs in identified subset from original Multi-MUM set

Recursive Anchoring



Do recursive anchoring with higher sensitivity

Gapped Alignment



Generate progressive alignments for each of LCBs by Clustalw with single guide tree constructed before

A Brief Overview of ProgressiveMauve

Find multi-MUMs (or LMA) (**Spaced Seeds**)

↓

Use the multi-MUMs to calculate a phylogenetic guide tree

↓

Select a subset of the multi-MUM to use as anchors, then partitioned into collinear groups call LCBs (**sum-of-pairs breakpoint score instead of weight**)

↓

Perform recursive anchoring to identify additional alignment anchors within and outside each LCBs

These steps are executed in pairwise according to guide tree

↓

Perform profile-profile alignment of each LCB using guide tree (**MUSCLE instead of Clustalw**)

↓

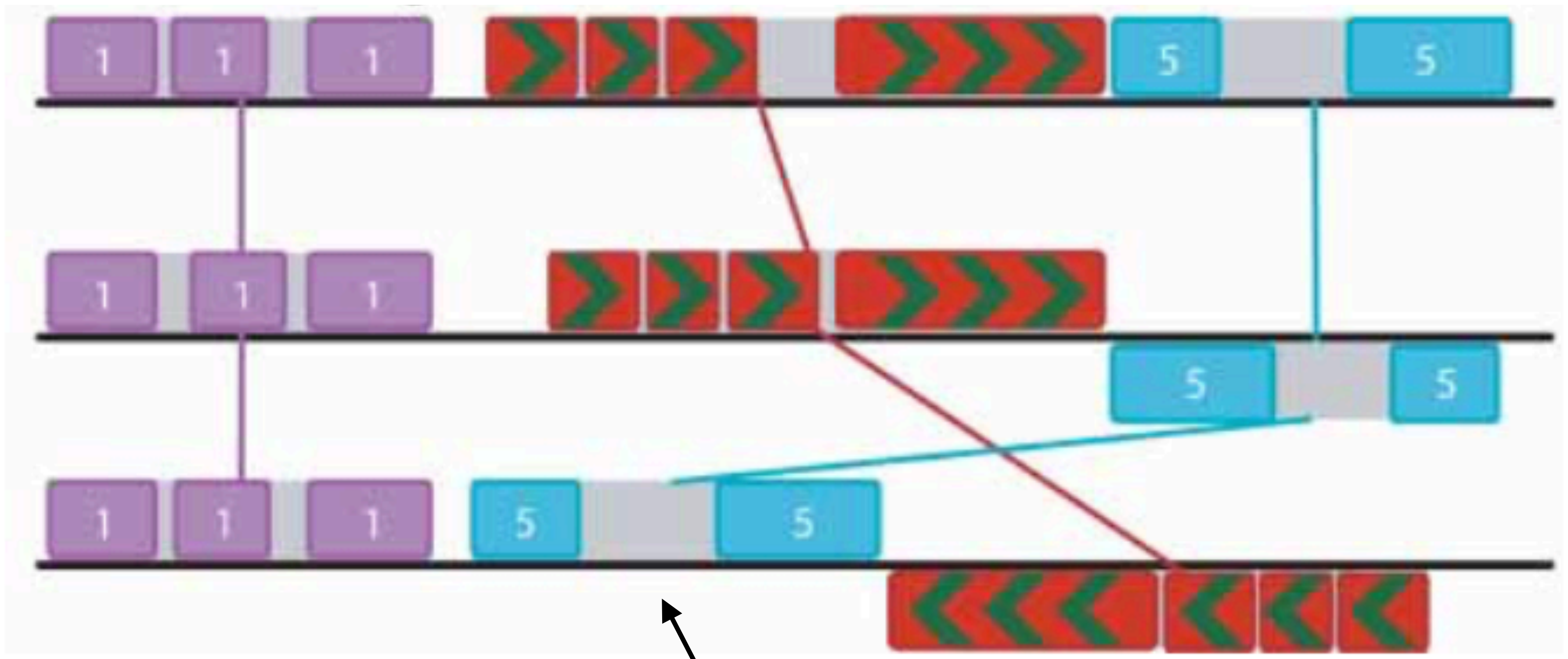
Rejecting alignments of unrelated sequences with a homology HMM

Sum-of-pairs Breakpoint Score

$$\underbrace{\Lambda(\mathcal{L}_{ij}) = -\beta}_{\text{Separation penalty}} \underbrace{\|\mathcal{L}_{ij}\|}_{\text{Number of LCBs}} + \underbrace{\sum_{m \in \mathcal{A}_{loc}} S(\pi_{ij}(m))}_{\text{Sum of scores of all LMAs in LCBs calculated based on HODX70}}$$

High sum of pair score means longer and more LMAs in LCBs with fewer breakpoints

Rejecting Alignments of Unrelated Sequences with a Homology HMM



Sequences here may be unrelated

Strength in Progressive Mauve

Works better on more divergent genomes (Spaced Seeds)

Large region of shared by subset genomes can be aligned
(LCBs are pair wisely identified)

More Accurate (Sum of pair breakpoint score, alignment refinement and back-end filtering)

Applied to a much larger number of genomes (Faster greedy algorithm in LCB identification)

Manual adjustment of the alignment scoring parameters is usually not necessary

Implementation Time

Platform, Compiler and Other Software Required

Mauve supports Windows, Linux and Mac OS X systems

Java 1.4 is required, while it has been already installed for most of the system (i.e Mac OS X, Fedora, Red Hat etc.)

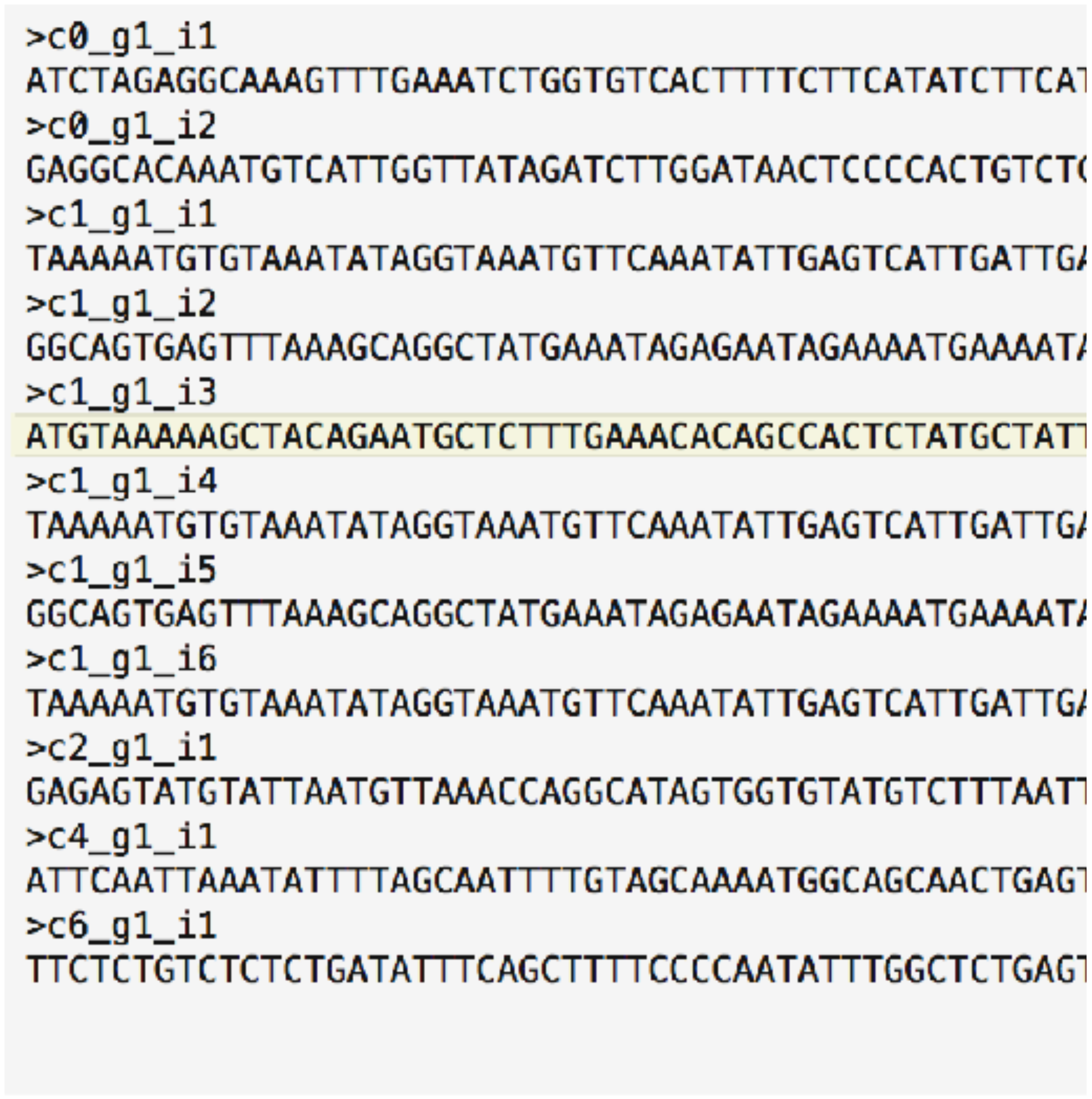
The Windows version of Mauve includes the Java installer for 32-bit windows systems, while ther systems Java may need to be installed separately.

Installation

Mauve provide easy-to-install installation package for Windows, Linux and Mac OS X systems.

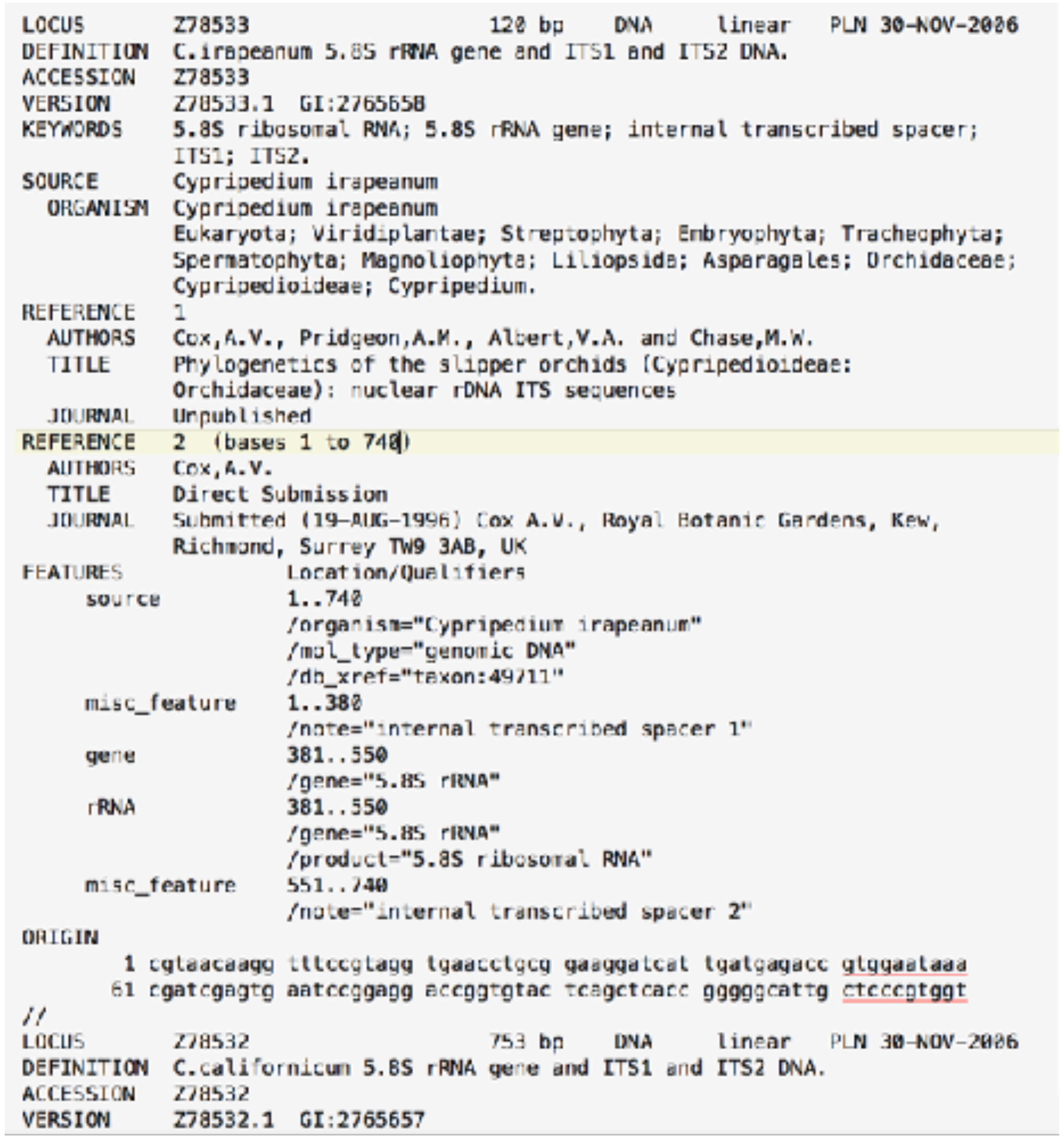
Other Unix-like operating systems, you can build from source.

Common Input Files



```
>c0_g1_i1
ATCTAGAGGCAAAGTTTGAAATCTGGTGTCACTTTTCTTCATATCTTCAT
>c0_g1_i2
GAGGCACAAATGTCATTGGTTATAGATCTTGGATAACTCCCCACTGTCTC
>c1_g1_i1
TAAAAATGTGTAAATATAGGTAAATGTTCAAATATTGAGTCATTGATTGA
>c1_g1_i2
GGCAGTGAGTTTAAAGCAGGCTATGAAATAGAGAATAGAAAATGAAAATA
>c1_g1_i3
ATGTAAAAAGCTACAGAATGCTCTTTGAAACACAGCCACTCTATGCTATT
>c1_g1_i4
TAAAAATGTGTAAATATAGGTAAATGTTCAAATATTGAGTCATTGATTGA
>c1_g1_i5
GGCAGTGAGTTTAAAGCAGGCTATGAAATAGAGAATAGAAAATGAAAATA
>c1_g1_i6
TAAAAATGTGTAAATATAGGTAAATGTTCAAATATTGAGTCATTGATTGA
>c2_g1_i1
GAGAGTATGTATTAATGTTAAACCAGGCATAGTGGTGTATGTCTTTAATT
>c4_g1_i1
ATTCAATTAAATATTTTAGCAATTTTGTAGCAAAATGGCAGCAACTGAGT
>c6_g1_i1
TTCTCTGTCTCTCTGATATTTCAGCTTTTCCCCAATATTTGGCTCTGAGT
```

Fasta format



```
LOCUS       Z78533                   120 bp    DNA     linear   PLN 30-NOV-2006
DEFINITION  C.irapeanum 5.8S rRNA gene and ITS1 and ITS2 DNA.
ACCESSION   Z78533
VERSION     Z78533.1  GI:2765658
KEYWORDS    5.8S ribosomal RNA; 5.8S rRNA gene; internal transcribed spacer;
            ITS1; ITS2.
SOURCE      Cypripedium irapeanum
  ORGANISM  Cypripedium irapeanum
            Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
            Spermatophyta; Magnoliophyta; Liliopsida; Asparagales; Orchidaceae;
            Cypripedioideae; Cypripedium.
REFERENCE   1
  AUTHORS   Cox,A.V., Pridgeon,A.M., Albert,V.A. and Chase,M.W.
  TITLE     Phylogenetics of the slipper orchids (Cypripedioideae:
            Orchidaceae): nuclear rDNA ITS sequences
  JOURNAL   Unpublished
REFERENCE   2  (bases 1 to 740)
  AUTHORS   Cox,A.V.
  TITLE     Direct Submission
  JOURNAL   Submitted (19-AUG-1996) Cox A.V., Royal Botanic Gardens, Kew,
            Richmond, Surrey TW9 3AB, UK
FEATURES             Location/Qualifiers
     source          1..740
                     /organism="Cypripedium irapeanum"
                     /mol_type="genomic DNA"
                     /db_xref="taxon:49711"
     misc_feature    1..380
                     /note="internal transcribed spacer 1"
     gene            381..550
                     /gene="5.8S rRNA"
     rRNA            381..550
                     /gene="5.8S rRNA"
                     /product="5.8S ribosomal RNA"
     misc_feature    551..740
                     /note="internal transcribed spacer 2"
ORIGIN
        1 cgtaacaagg tttccgtagg tgaacctgcg gaaggatcat tgatgagacc gtggaataaa
       61 cgatcgagtg aatccggagg accggtgtac tcagctcacc gggggcattg ctcccgtggt
//
LOCUS       Z78532                   753 bp    DNA     linear   PLN 30-NOV-2006
DEFINITION  C.californicum 5.8S rRNA gene and ITS1 and ITS2 DNA.
ACCESSION   Z78532
VERSION     Z78532.1  GI:2765657
```

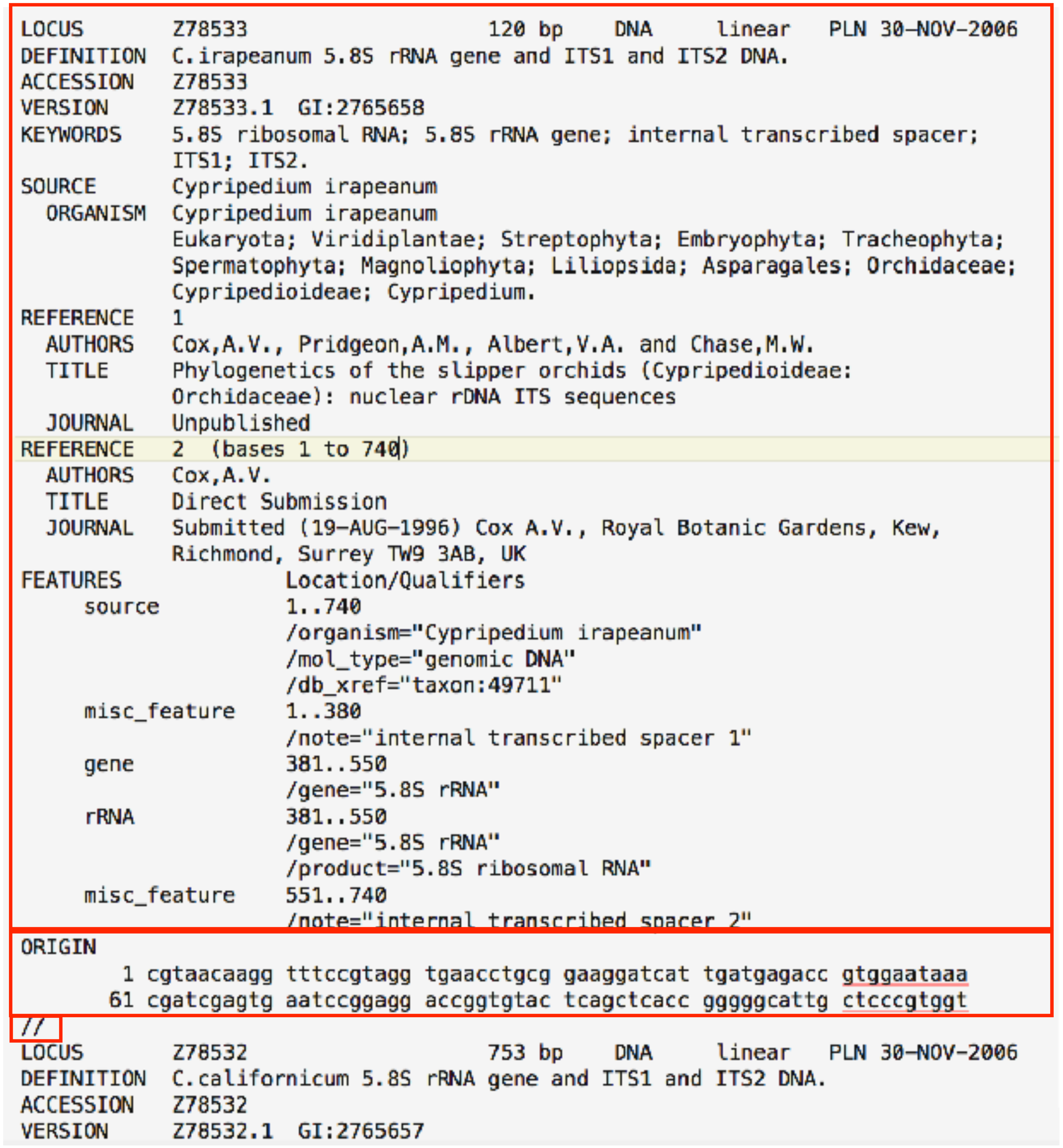
Genbank format

Genbank format

Lots of description

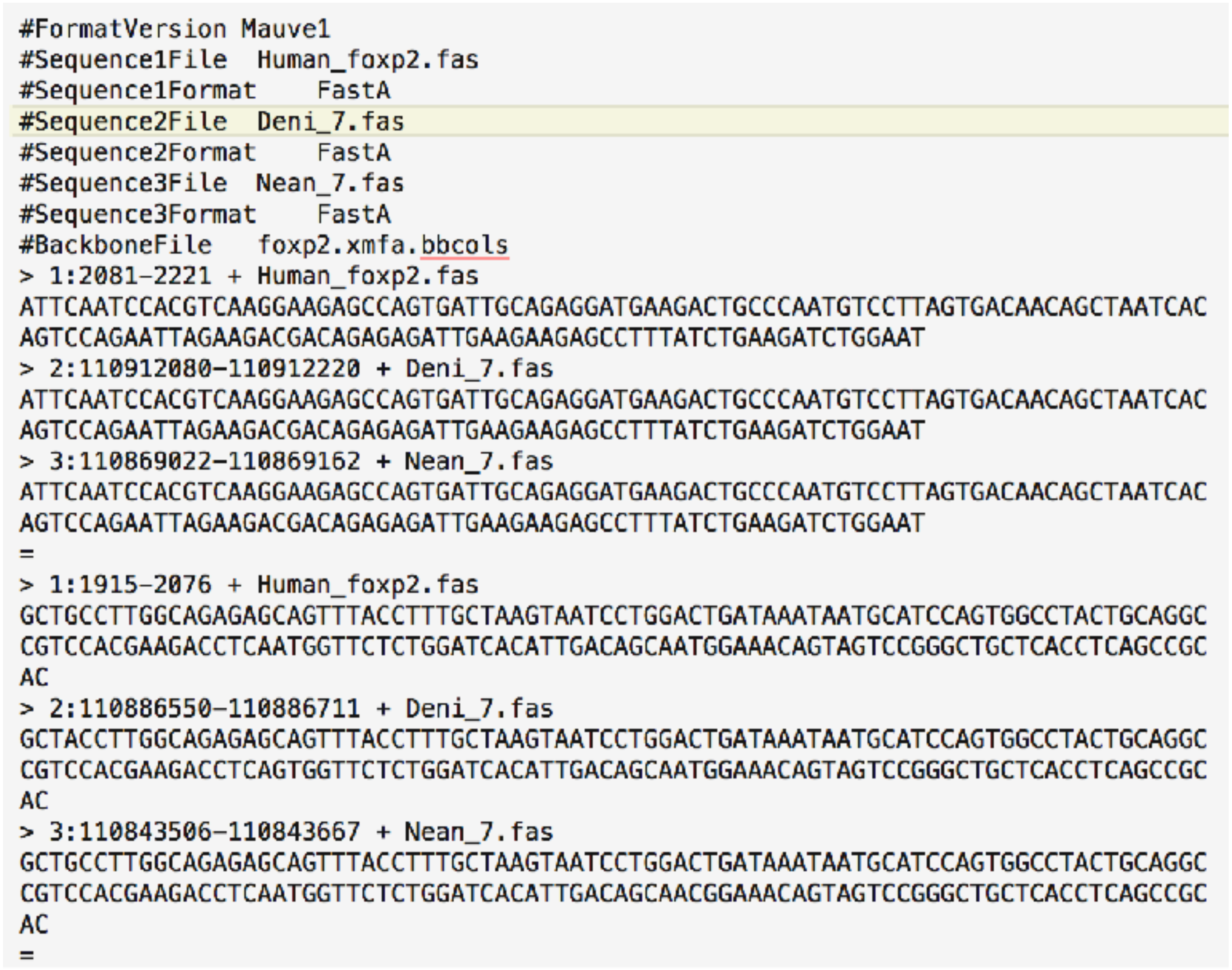
Sequence

“//” separate the block



```
LOCUS       Z78533                   120 bp    DNA     linear   PLN 30-NOV-2006
DEFINITION  C.irapeanum 5.8S rRNA gene and ITS1 and ITS2 DNA.
ACCESSION   Z78533
VERSION     Z78533.1  GI:2765658
KEYWORDS    5.8S ribosomal RNA; 5.8S rRNA gene; internal transcribed spacer;
            ITS1; ITS2.
SOURCE      Cypripedium irapeanum
  ORGANISM  Cypripedium irapeanum
            Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
            Spermatophyta; Magnoliophyta; Liliopsida; Asparagales; Orchidaceae;
            Cypripedioideae; Cypripedium.
REFERENCE   1
  AUTHORS   Cox,A.V., Pridgeon,A.M., Albert,V.A. and Chase,M.W.
  TITLE     Phylogenetics of the slipper orchids (Cypripedioideae:
            Orchidaceae): nuclear rDNA ITS sequences
  JOURNAL   Unpublished
REFERENCE   2  (bases 1 to 740)
  AUTHORS   Cox,A.V.
  TITLE     Direct Submission
  JOURNAL   Submitted (19-AUG-1996) Cox A.V., Royal Botanic Gardens, Kew,
            Richmond, Surrey TW9 3AB, UK
FEATURES             Location/Qualifiers
     source          1..740
                     /organism="Cypripedium irapeanum"
                     /mol_type="genomic DNA"
                     /db_xref="taxon:49711"
     misc_feature    1..380
                     /note="internal transcribed spacer 1"
     gene            381..550
                     /gene="5.8S rRNA"
     rRNA            381..550
                     /gene="5.8S rRNA"
                     /product="5.8S ribosomal RNA"
     misc_feature    551..740
                     /note="internal transcribed spacer 2"
ORIGIN
        1 cgtaacaagg tttccgtagg tgaacctgcg gaaggatcat tgatgagacc gtggaataaa
       61 cgatcgagtg aatccggagg accggtgtac tcagctcacc gggggcattg ctcccgtggt
//
LOCUS       Z78532                   753 bp    DNA     linear   PLN 30-NOV-2006
DEFINITION  C.californicum 5.8S rRNA gene and ITS1 and ITS2 DNA.
ACCESSION   Z78532
VERSION     Z78532.1  GI:2765657
```

Common Output Files Format



```
#FormatVersion Mauve1
#Sequence1File  Human_foxp2.fas
#Sequence1Format    FastA
#Sequence2File  Deni_7.fas
#Sequence2Format    FastA
#Sequence3File  Nean_7.fas
#Sequence3Format    FastA
#BackboneFile   foxp2.xmfa.bbcols
> 1:2081-2221 + Human_foxp2.fas
ATTCAATCCACGTCAAGGAAGAGCCAGTGATTGCAGAGGATGAAGACTGCCCAATGTCCTTAGTGACAACAGCTAATCAC
AGTCCAGAATTAGAAGACGACAGAGAGATTGAAGAAGAGCCTTTATCTGAAGATCTGGAAT
> 2:110912080-110912220 + Deni_7.fas
ATTCAATCCACGTCAAGGAAGAGCCAGTGATTGCAGAGGATGAAGACTGCCCAATGTCCTTAGTGACAACAGCTAATCAC
AGTCCAGAATTAGAAGACGACAGAGAGATTGAAGAAGAGCCTTTATCTGAAGATCTGGAAT
> 3:110869022-110869162 + Nean_7.fas
ATTCAATCCACGTCAAGGAAGAGCCAGTGATTGCAGAGGATGAAGACTGCCCAATGTCCTTAGTGACAACAGCTAATCAC
AGTCCAGAATTAGAAGACGACAGAGAGATTGAAGAAGAGCCTTTATCTGAAGATCTGGAAT
=
> 1:1915-2076 + Human_foxp2.fas
GCTGCCTTGGCAGAGAGCAGTTTACCTTTGCTAAGTAATCCTGGACTGATAAATAATGCATCCAGTGGCCTACTGCAGGC
CGTCCACGAAGACCTCAATGGTTCTCTGGATCACATTGACAGCAATGGAAACAGTAGTCCGGGCTGCTCACCTCAGCCGC
AC
> 2:110886550-110886711 + Deni_7.fas
GCTACCTTGGCAGAGAGCAGTTTACCTTTGCTAAGTAATCCTGGACTGATAAATAATGCATCCAGTGGCCTACTGCAGGC
CGTCCACGAAGACCTCAGTGGTTCTCTGGATCACATTGACAGCAATGGAAACAGTAGTCCGGGCTGCTCACCTCAGCCGC
AC
> 3:110843506-110843667 + Nean_7.fas
GCTGCCTTGGCAGAGAGCAGTTTACCTTTGCTAAGTAATCCTGGACTGATAAATAATGCATCCAGTGGCCTACTGCAGGC
CGTCCACGAAGACCTCAATGGTTCTCTGGATCACATTGACAGCAACGGAAACAGTAGTCCGGGCTGCTCACCTCAGCCGC
AC
=
```

XMFA format

Common Output Files Format

bbcols files contain all backbone entries

Backbone are regions in the correct alignment containing >50 gap-free columns without stretches of 50 or more consecutive gaps in any single genome sequence.

Basic Pipeline

1. Generate genomic multiple alignment:

```
progressiveMauve --output=full_alignment.xmfa genome1.fas genome2.fas
genome3.fas genome4.fas
```

2. Select conserved backbone alignment:

```
stripSubsetLCBs full_alignment.xmfa full_alignment.xmfa.bbcols
filtered_full_alignment.xmfa length number_of_seq
```

Following the introduction of bioperl