Assembly Pipeline Introduction and Usage Example

Hao Yuan

Where are we now?

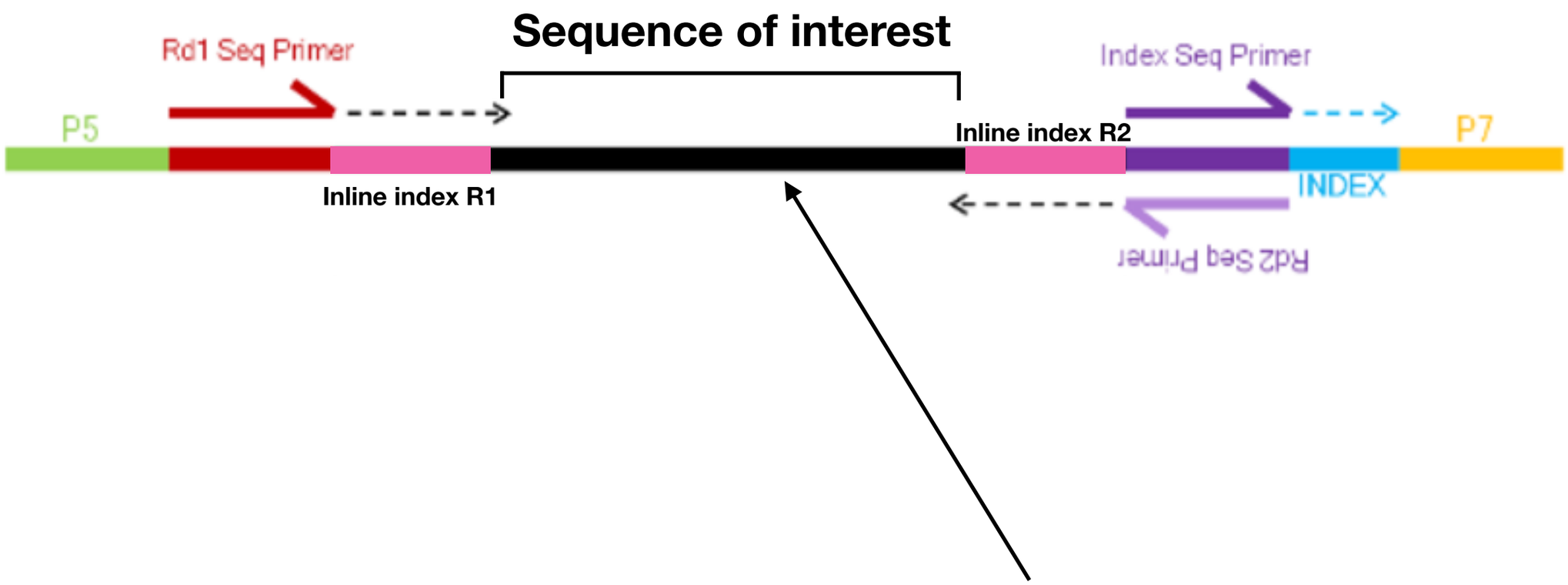
ATCG…..

Exhausting experiment

Sequencing

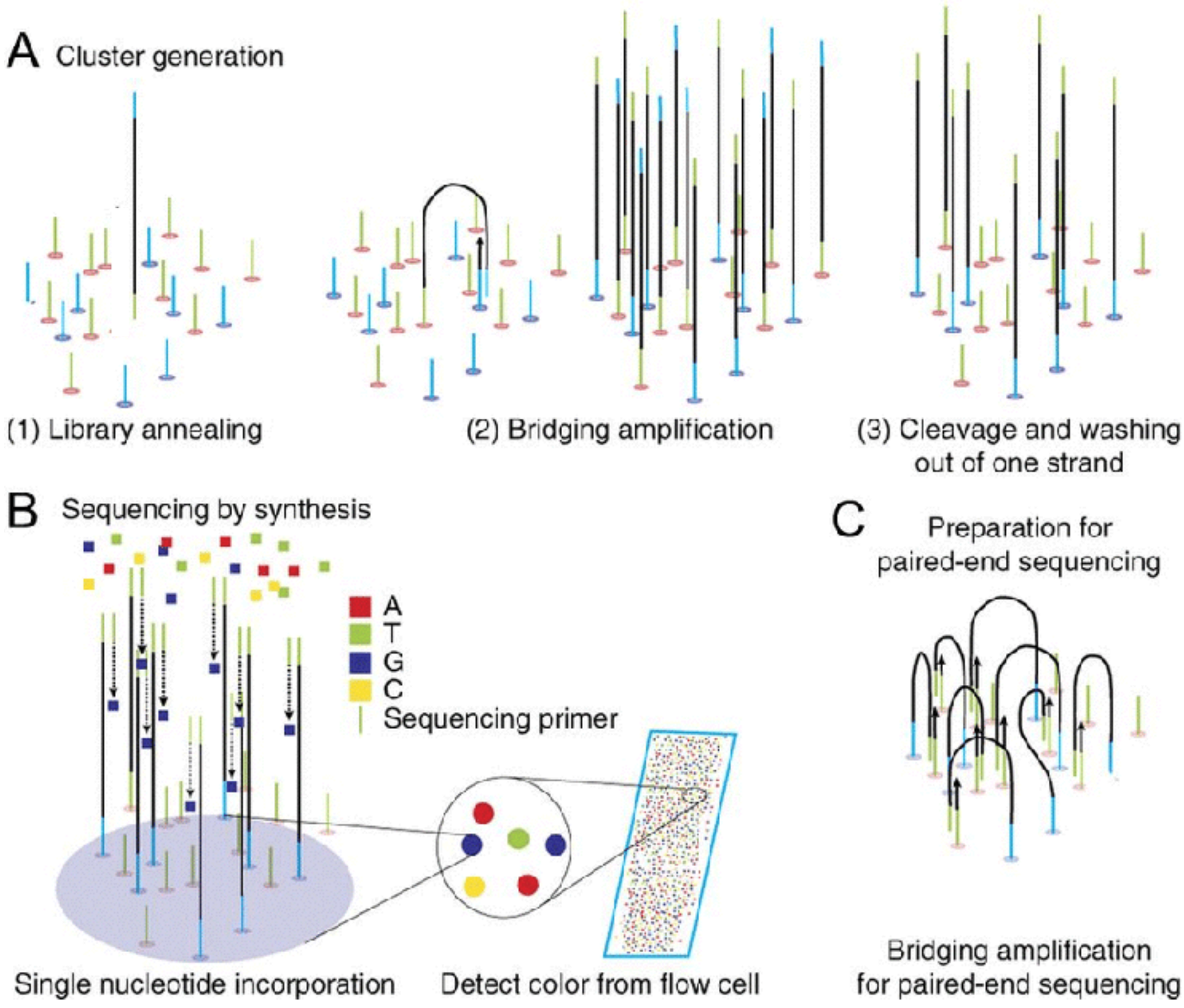
Analysis our data

Structure of prepared library

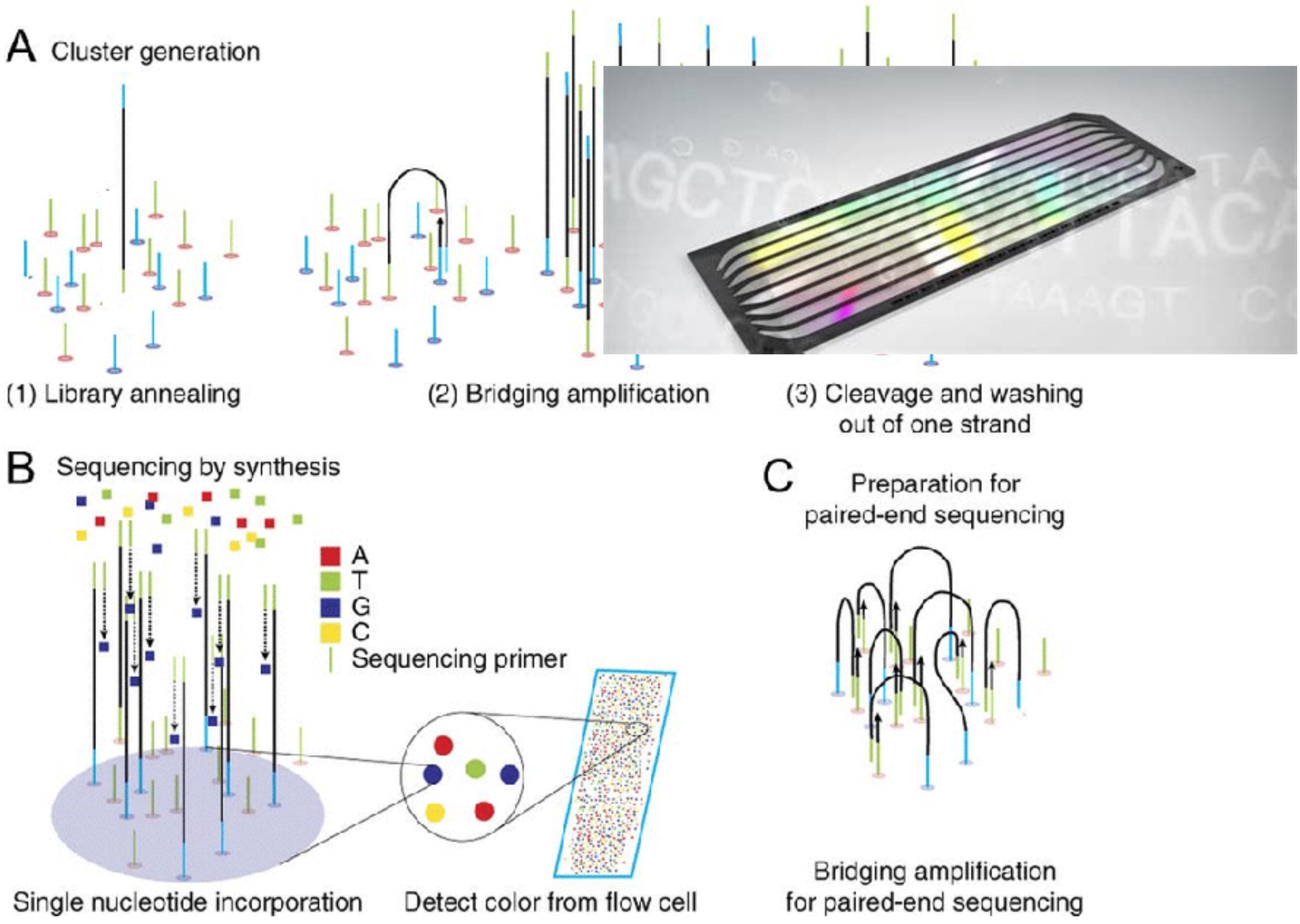


Length of this part is sometimes called "insert size", which totally depends on the length of DNA after shearing

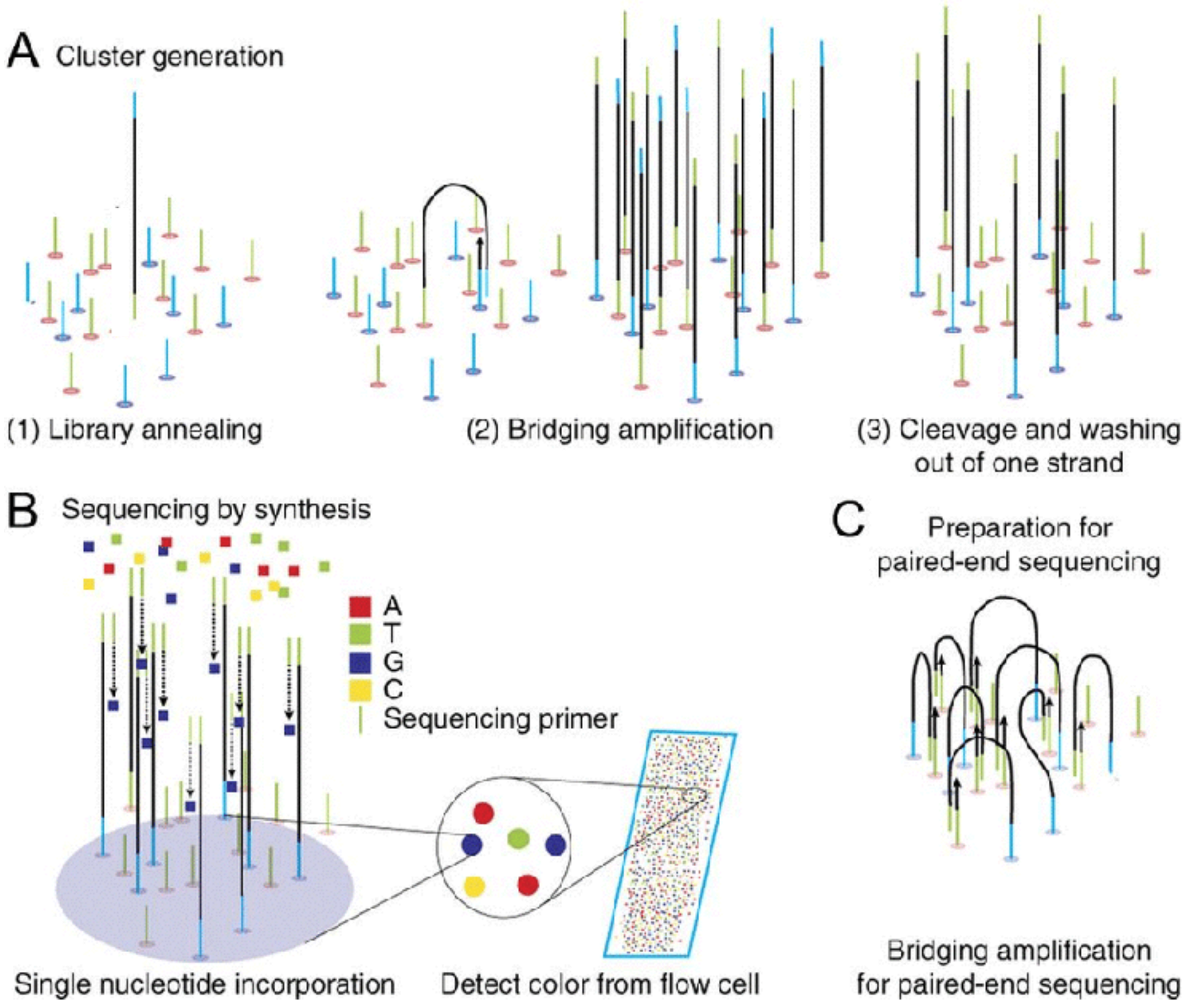
Pair-end Sequencing



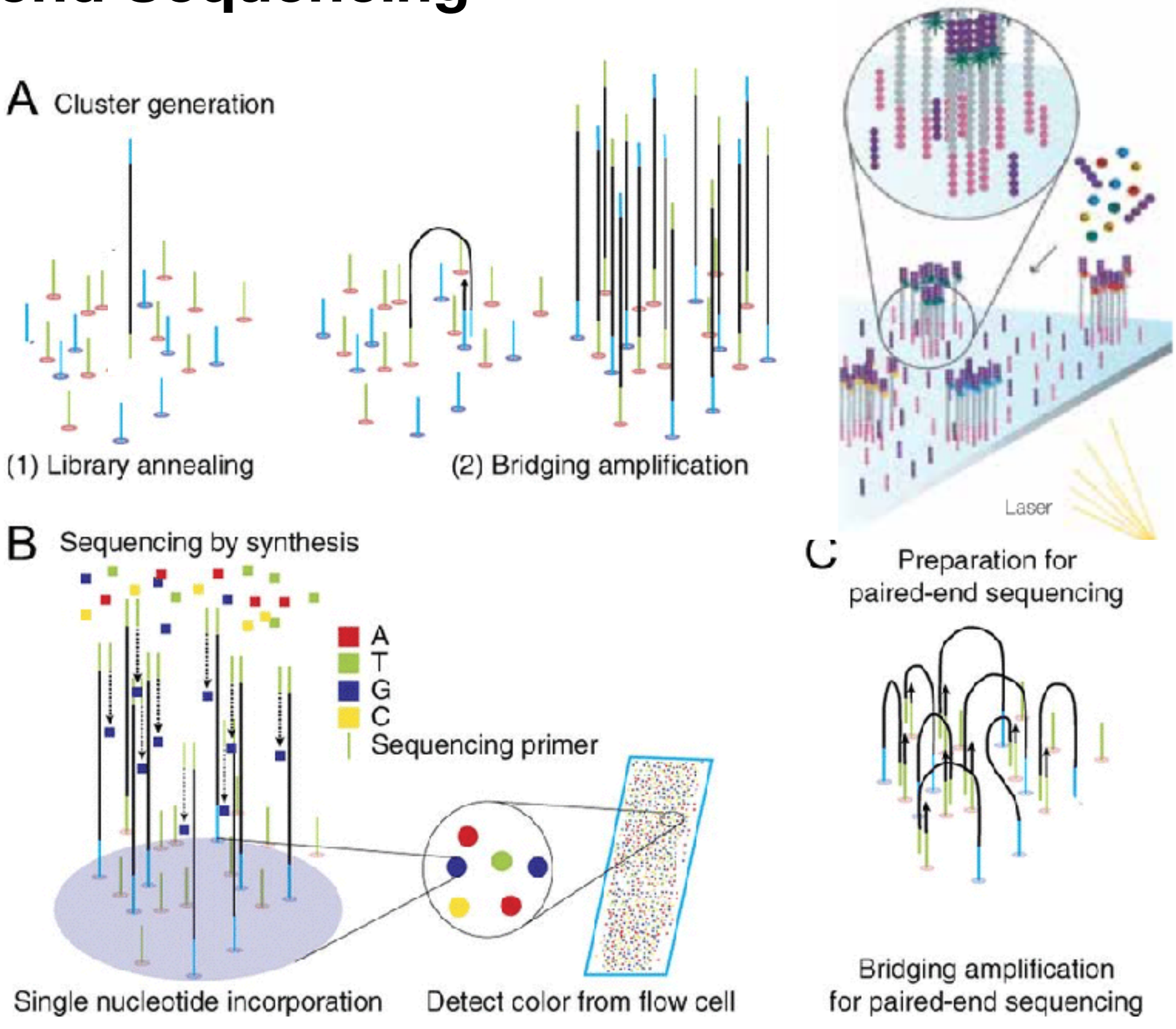
Pair-end Sequencing



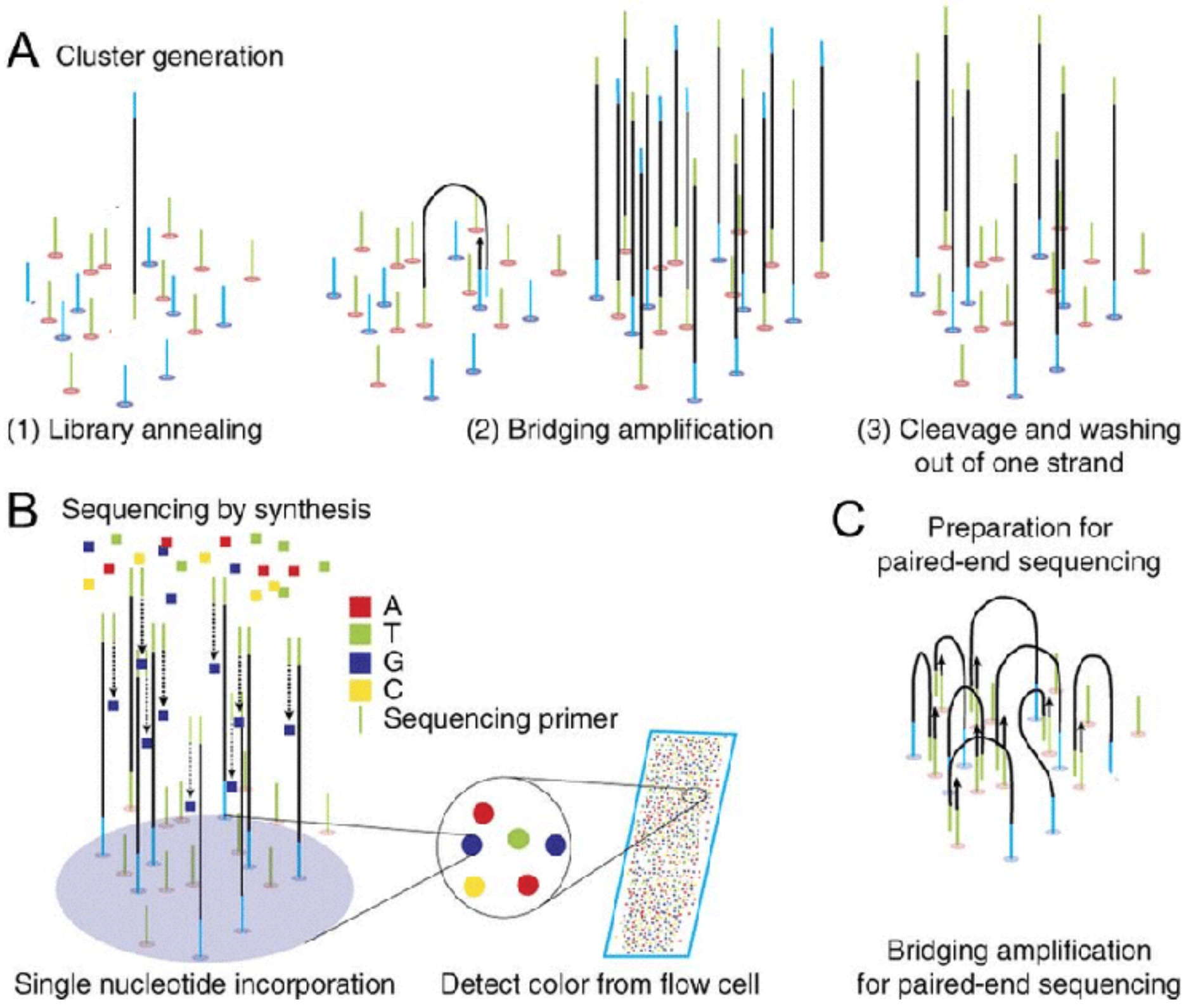
Pair-end Sequencing



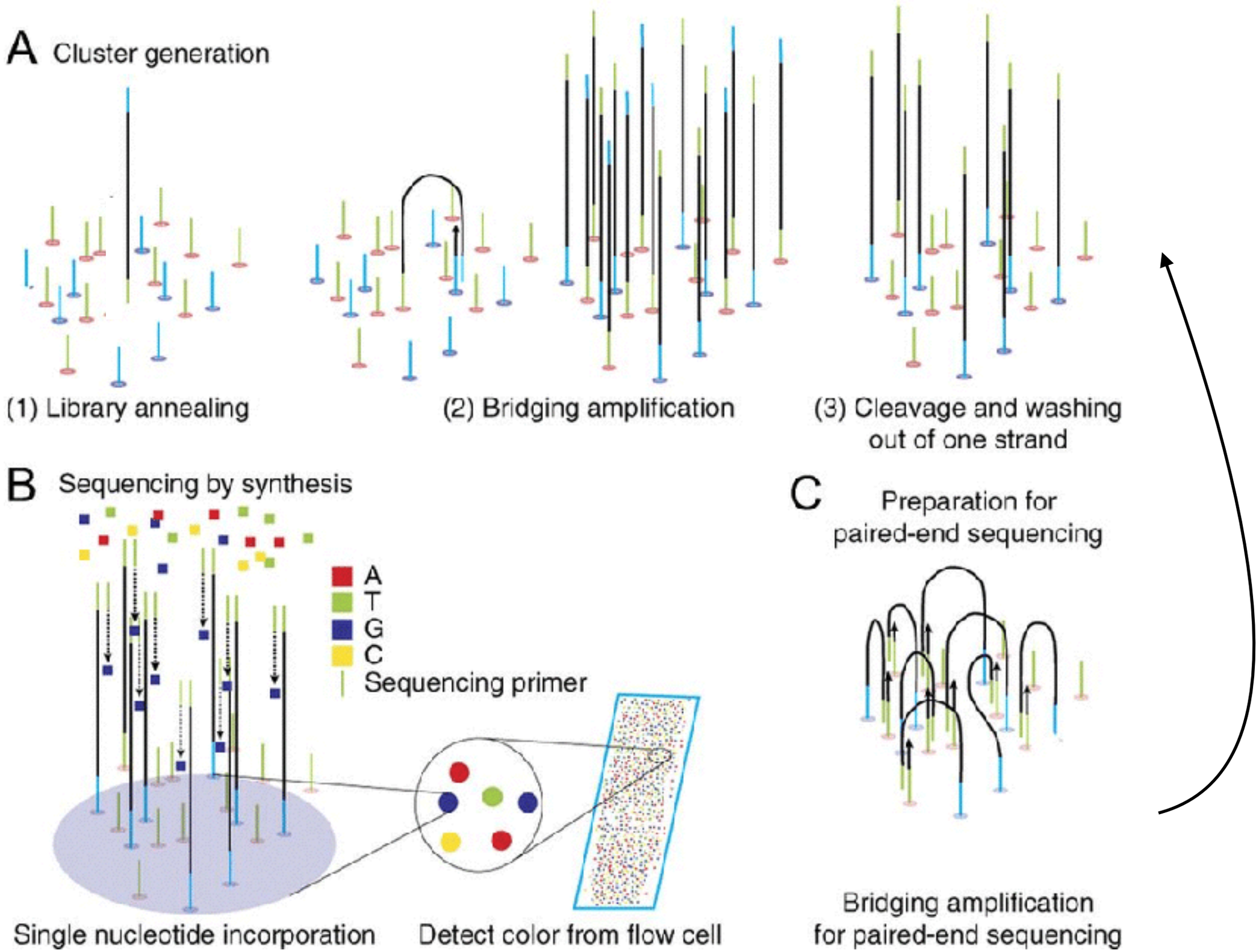
Pair-end Sequencing



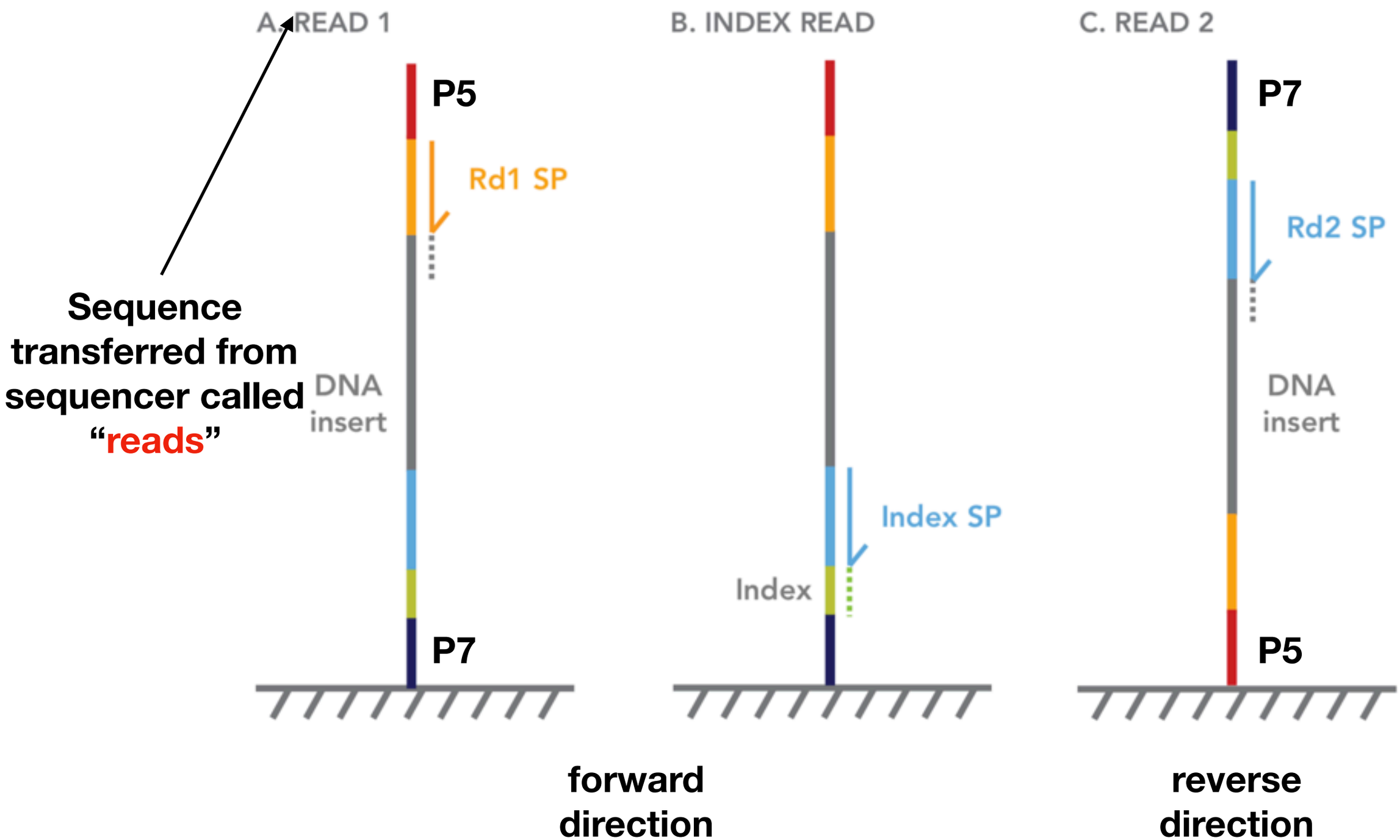
Pair-end Sequencing



Pair-end Sequencing

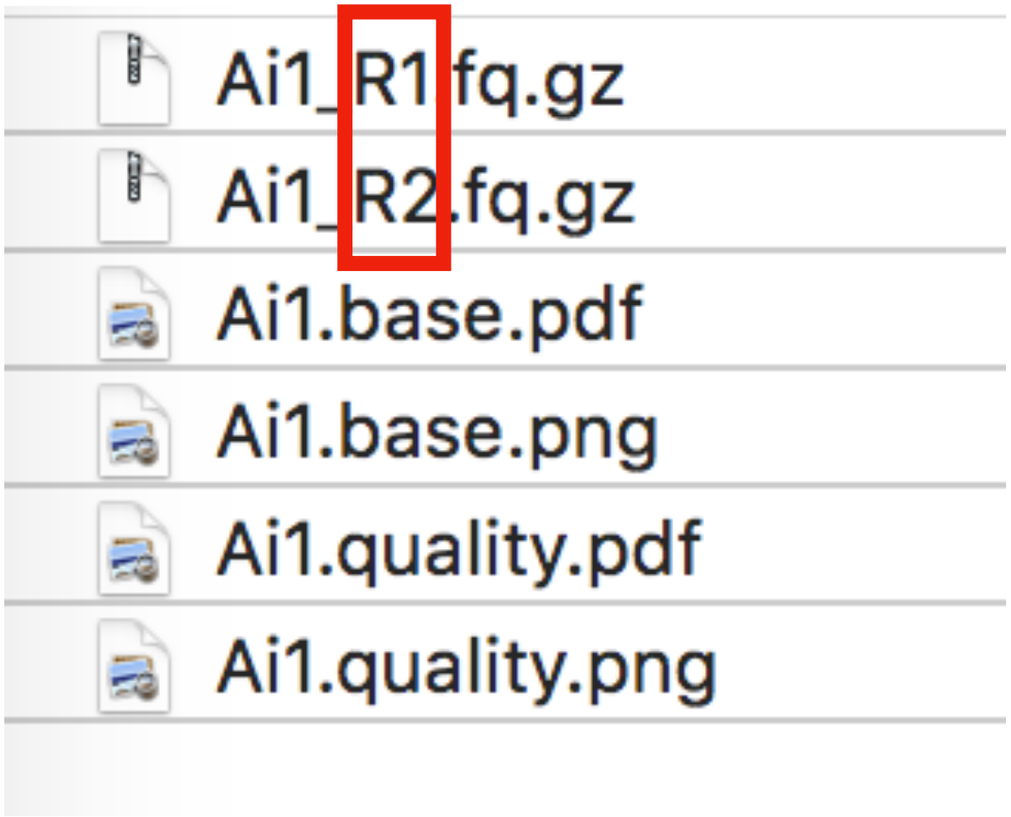


Pair-end Sequencing



What comes from the sequencer

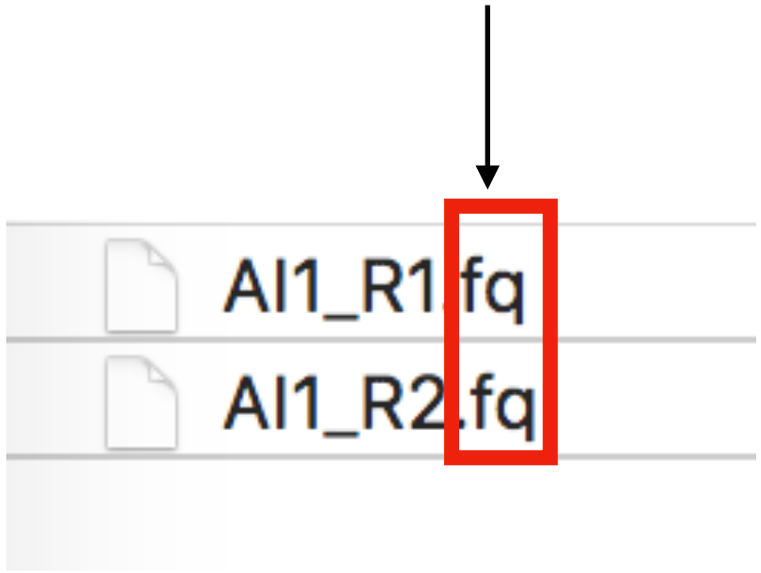
File suffix is “fq” or “fastq”



Reads

Bases composition

Quality of Bases



Assigned (more professionally called “demultiplexed ”) reads according to index close to P7

Expanded file

How reads in xxx_R1.fq looks like

header

index sequence

sequence

a “plus”

quality (phred 33)

```
@E00487:317:HJLJHCCXY:5:1101:10734:2012 1:N:0:CGTCATAA
ACTACACTTTCTTTAGAAAATGCATTTTCTAGTTTATATTGATATCATTAACATTTTGCCTAAACTAAACTGGACTAACC
CTAATCAAATCATAAAATCTAACAAGTTACCTAGTTTTTTGACCCGACATTAAAGGTGAGTCAA
+
FAJJJJJJJJJJJJFF-<FFJJJ<JJJJJJJJJJJAFJJJJJJJJJJJJJJJJJJJFJJJF<JFJ<FJFJJJFJFJJJ7
JFA-FAJFJJJJJJFFJJFJ<FFJJJJJJAJJFJFFJJJJAAJ<FJ<JJJJJJJJJJJFFJFF
```

How reads in xxx_R2.fq looks like

header

index sequence

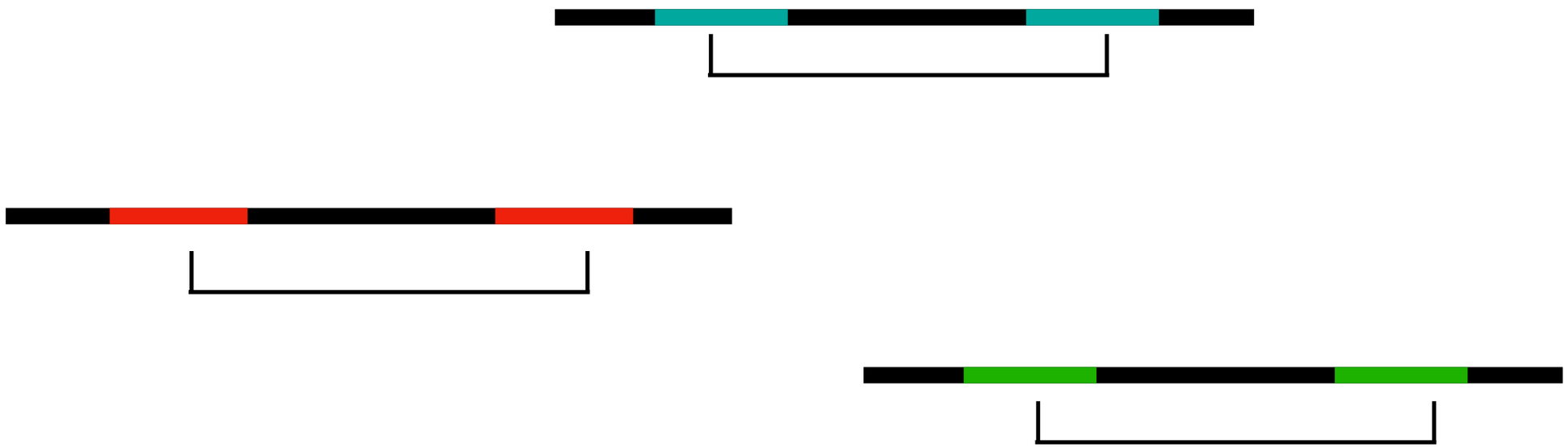
sequence

a “plus”

quality (phred 33)

```
@E00487:317:HJLJHCCXY:5:1101:20943:2012 2:N:0:CGTCATAA
AAGTTGACGGTGCTAGCAAGCTAAAGCTACCCGGAGATATGGAGGCAACTGAAGCCAACATTCTTGCGGAGCTGAAAAAA
GTACAGGATAATTTACAAGGTGAAATAGCGGACACGATTGGACTTGTGCAAACAGAAATGTCG
+
FFFFJJFFJJJJJJA-AJJF-AFA<FJFJ<<AFJJ7-<F-FJJFJ77--<F<FJ<-AF-<-<<-JJ77FJJ<J--AA---
-<7AFJJJJJ-7--7-FJ--<AFAJJJF-AF-77FAJ--7-7---7AF---7-<-AF7A7-7-
```

What’s paired reads



Reads originated from the same molecular before bridge amplification called paired reads

Paired reads in xxx_R1.fq and xxx_R2.fq

xxx_R1.fq

```
@E00487:317:HJLJHCCXY:5:1101:10734:2012 1:N:0:CGTCATAA
ACTACACTTTCTTTAGAAAATGCATTTTCTAGTTTATATTGATATCATTAACATTTTGCCTAAACTAAACTGGACTAACC
CTAATCAAATCATAAAATCTAACAAGTTACCTAGTTTTTTGACCCGACATTAAAGGTGAGTCAA
+
FAJJJJJJJJJJJJFF-<FFJJJ<JJJJJJJJJJJAFJJJJJJJJJJJJJJJJJJJJFJJJF<JFJ<FJFJJJFJFJJJ7
JFA-FAJFJJJJJJFFJJFJ<FFJJJJJJAJJFJFFJJJJAAJ<FJ<JJJJJJJJJJJJFFJFF
```

xxx_R2.fq

```
@E00487:317:HJLJHCCXY:5:1101:20943:2012 2:N:0:CGTCATAA
AAGTTGACGGTGCTAGCAAGCTAAAGCTACCCGGAGATATGGAGGCAACTGAAGCCAACATTCTTGCGGAGCTGAAAAAA
GTACAGGATAATTTACAAGGTGAAATAGCGGACACGATTGGACTTGTGCAAACAGAAATGTCG
+
FFFFJJFFJJJJJJA-AJJF-AFA<FJFJ<<AFJJ7-<F-FJJFJ77--<F<FJ<-AF-<-<<-JJ77FJJ<J--AA---
-<7AFJJJJJ-7--7-FJ--<AFAJJJF-AF-77FAJ--7-7---7AF---7-<-AF7A7-7-
```

Paired reads have unique id in their corresponding file and exactly the same name

Compare reads in xxx_R1.fq and xxx_R2.fq

xxx_R1.fq

```
@E00492:247:HFMH3CCXY:8:1101:18436:2909 1:N:0:CTGCAATG
CTCAAAAACAACAATAATCTATTATCACAACCAAAGCAACAGCATACATATAATATCAATGCTTCTTTATCTCGATGTGAATAATAAGAAGTCAATGGGAACTTACCTTT
+
FJJJJAJJJJJFJJ<-FJJJFJJJJJJFJJJJFJJFJJJJJJJAJJJJJJJJJJJFJFJJ<JJJJJJFFAFJF77-A<AJJJJJAJJJJJFFJJJ7<F-7FJFJ-AJF-F
@E00492:247:HFMH3CCXY:8:1101:6725:3401 1:N:0:CTGCAATG
AATTCAGCGACTTGGTTCACACAATCCTCCATCCCTCCTTCCTTTACCACGATGTTTGATGATGCTGCAGGTACAGCTGCTGAAGAGACACGAGCTGTGACCAGTAACTA
+
7JJJAJJJ-FJJJJJFJJJJAJFJFJA-FFJJFJFJJJAAJJF7AJJJJJJJF7A7JJJJFJJJJFJJJAFJJAJFFFJJJ7JFFJJFJJJJJF<-----77---7--77
@E00492:247:HFMH3CCXY:8:1101:8745:3841 1:N:0:CTGCAATG
AAATGTAAGGTCAAACTACAACCTTGTCTATGAAGCCGGTGTGCCATTTGCGTAGCTTCCTGTTTTCGGTGGAGAGTTTCTCAGCAGCAGACTGCAGCTTGGCATTGTCA
+
JJJJJJJJJJJJJJFJJJFFJJJJFJJJJJJJJJJFAFJJJJJ<FJJJJFJJJJFA<FJ-<AAJFJJJF-7A-AJJ-7JA7FJJFJJ7JFJJJJJ7J7777--7-<A<7A
```

xxx_R2.fq

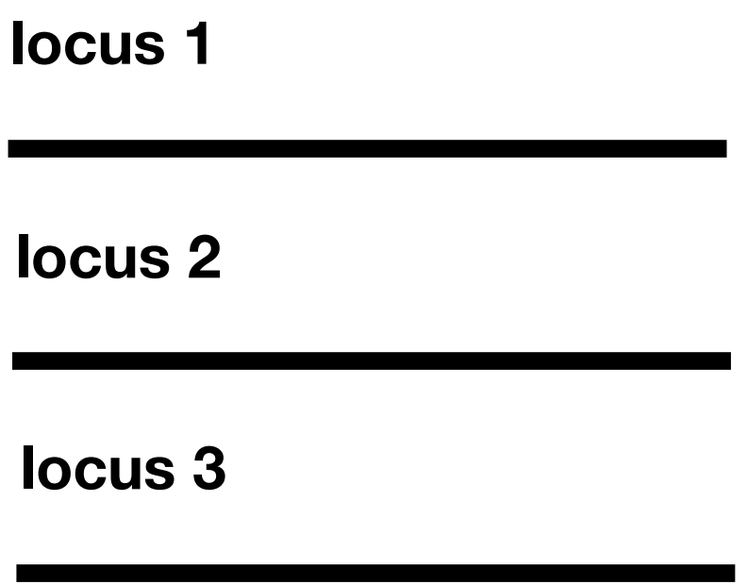
```
@E00492:247:HFMH3CCXY:8:1101:18436:2909 2:N:0:CTGCAATG
ATAATGGTTTTTGACAAGTTTAGCTTAAAGGAGGAAAATTGCATGTGTTGTGATCTTGTCACTCTCTGTGTCTCCTAGACTCCTGCAGAATTGCAAGGAGGACAGGTTAGA
+
AJJ--7F<-<F---<F--FFJ7FJ-77AJ-AF-77-7F<-<<---F-7-AF--7---<----7AAA7-7-----7AJ-A-AA-A-AJF7A7--AAF7J<A7FJAJ--7A-F
@E00492:247:HFMH3CCXY:8:1101:6725:3401 2:N:0:CTGCAATG
GGTATCTTTAAAACACATTTAACATGCGATACATAACTTATAATTGGTCATGTTTGTCATACAAATTGAATTAGACAGTGAATCAAAGAAAATGTGCTTACACTGACAAGT
+
JAA7A--<FJJ--<JFJ7FF<JFJJ<-F-<JJ-FJF77F<FJJJFJJA-FJJJFJ<<JJ7JJFFF<F-FFA-A--7<-F-AFAJJA<FFJFJFFFF-JJFJF--A7--77-
@E00492:247:HFMH3CCXY:8:1101:8745:3841 2:N:0:CTGCAATG
AATGGTTGCCAAGCAACACAGAGACGAAGTGGCAAGTATGATTGAAGATTGGAGTGATATGAGTGATATTAACATTCAGCTGGGGGTGATTATTAACATTTAGCAGGATAT
+
FAF<-7FA<7-<7<<7FFJ7F<<--A--<AFF-7F-A<FF-<A<-FF7FF7AJF77<JF-<AF-J-F-AFAFJJJAF-7<AAAAAJAFJ<FFFJFJJJF<J<<A<<<-A-7
```

Paired reads are placed at the same line of its corresponding file

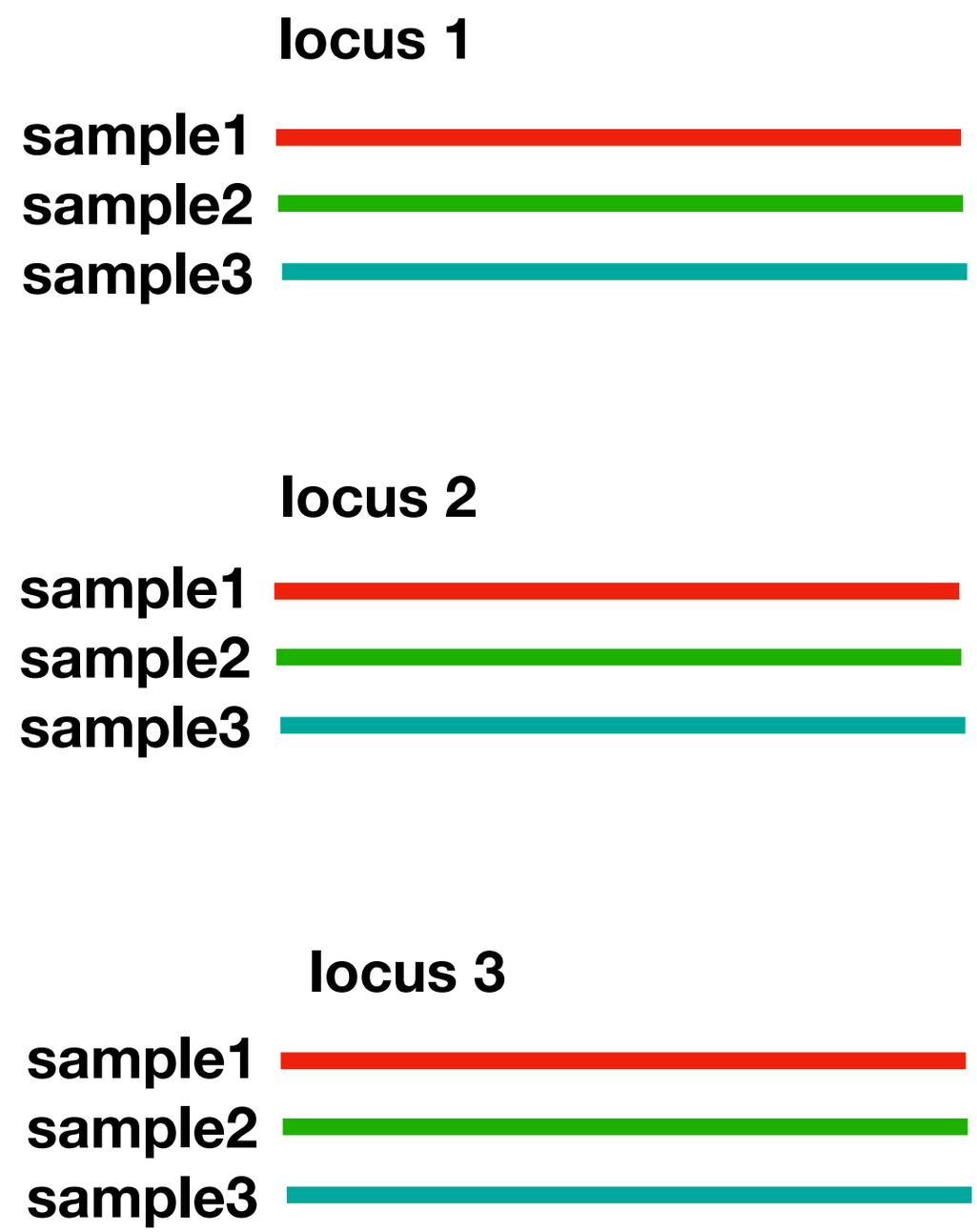
Let’s start analysis now!!

What's the goal of analysis?

The sequence of loci used to design the baits, we call it "reference"



Intact sequences of corresponding locus of each sample



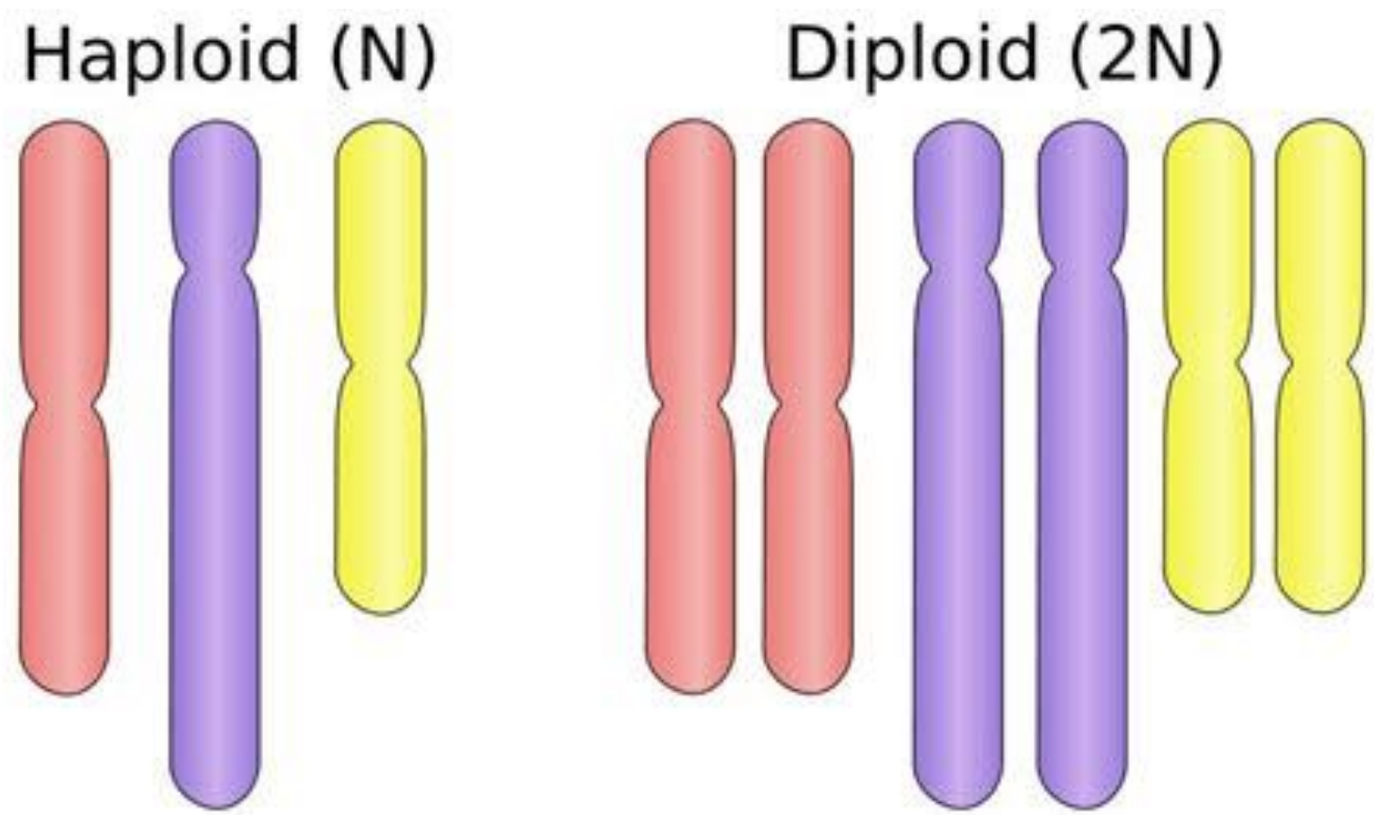
A locus is a fixed position on chromosome. In our lab, a locus is always an exon flanked by intron or UTR region

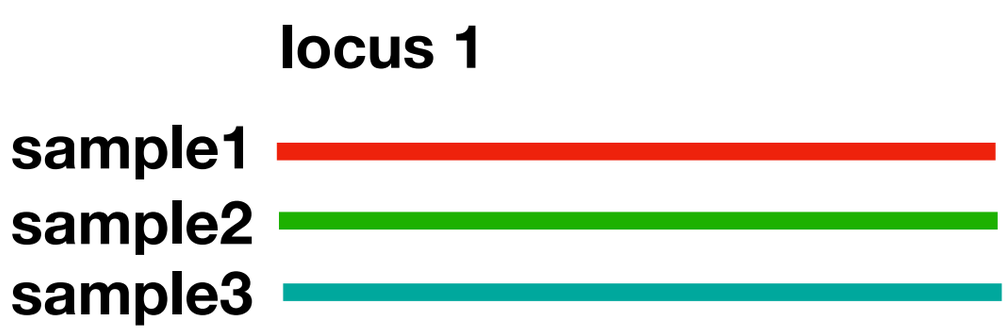
What’s the goal of analysis?

```
@E00492:247:HFMH3CCXY:8:1101:18436:2909 1:N:0:CTGCAATG
CTCAAAAACAACAATAATCTATTATCACAACCAAAGCAACAGCATACATATAATATCAATGCTTCTTTATCTCGATGTGAATAATAAGAAGTCAATGGGAACTTACCTTT
+
FJJJJAJJJJJFJJ<-FJJJFJJJJJJFJJJJFJJFJJJJJJJAJJJJJJJJJJJFJFJJ<JJJJJJFFAFJF77-A<AJJJJJAJJJJJFFJJJ7<F-7FJFJ-AJF-F
@E00492:247:HFMH3CCXY:8:1101:6725:3401 1:N:0:CTGCAATG
AATTCAGCGACTTGGTTCACACAATCCTCCATCCCTCCTTCCTTTACCACGATGTTTGATGATGCTGCAGGTACAGCTGCTGAAGAGACACGAGCTGTGACCAGTAACTA
+
7JJJAJJJ-FJJJJJFJJJJAJFJFJA-FFJJFJFJJJAAJJF7AJJJJJJJF7A7JJJJFJJJJFJJJAFJJAJFFFJJJ7JFFJJFJJJJJF<-----77---7--77
@E00492:247:HFMH3CCXY:8:1101:8745:3841 1:N:0:CTGCAATG
AAATGTAAGGTCAAACTACAACCTTGTCTATGAAGCCGGTGTGCCATTTGCGTAGCTTCCTGTTTTCGGTGGAGAGTTTCTCAGCAGCAGACTGCAGCTTGGCATTGTCA
+
JJJJJJJJJJJJJJFJJJFFJJJJFJJJJJJJJJJFAFJJJJJ<FJJJJFJJJJFA<FJ-<AAJFJJJF-7A-AJJ-7JA7FJJFJJ7JFJJJJJ7J7777--7-<A<7A
```

Short raw data

There’s only one sequence for each sample





3 steps to recover qualified assemblies from raw data

Data preparation

Assembling

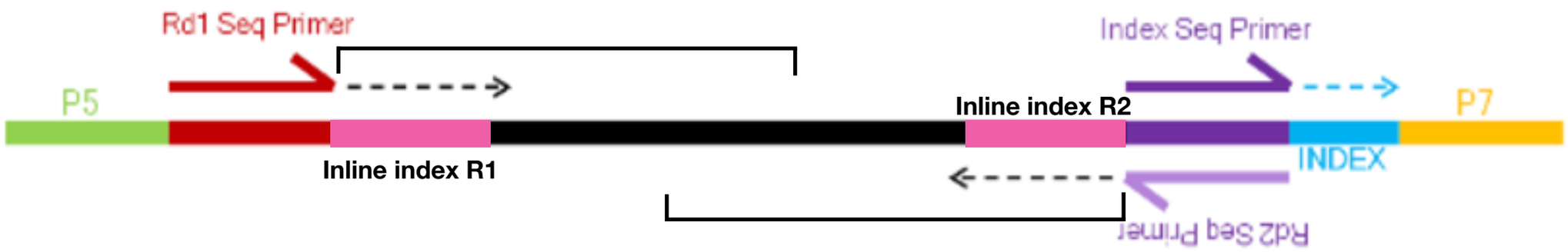
Further processing

Data preparation

Demultiplex reads according to inline index

Trim low quality bases and adaptor

Demultiplex reads according to inline index



Reads we got still includes inline index

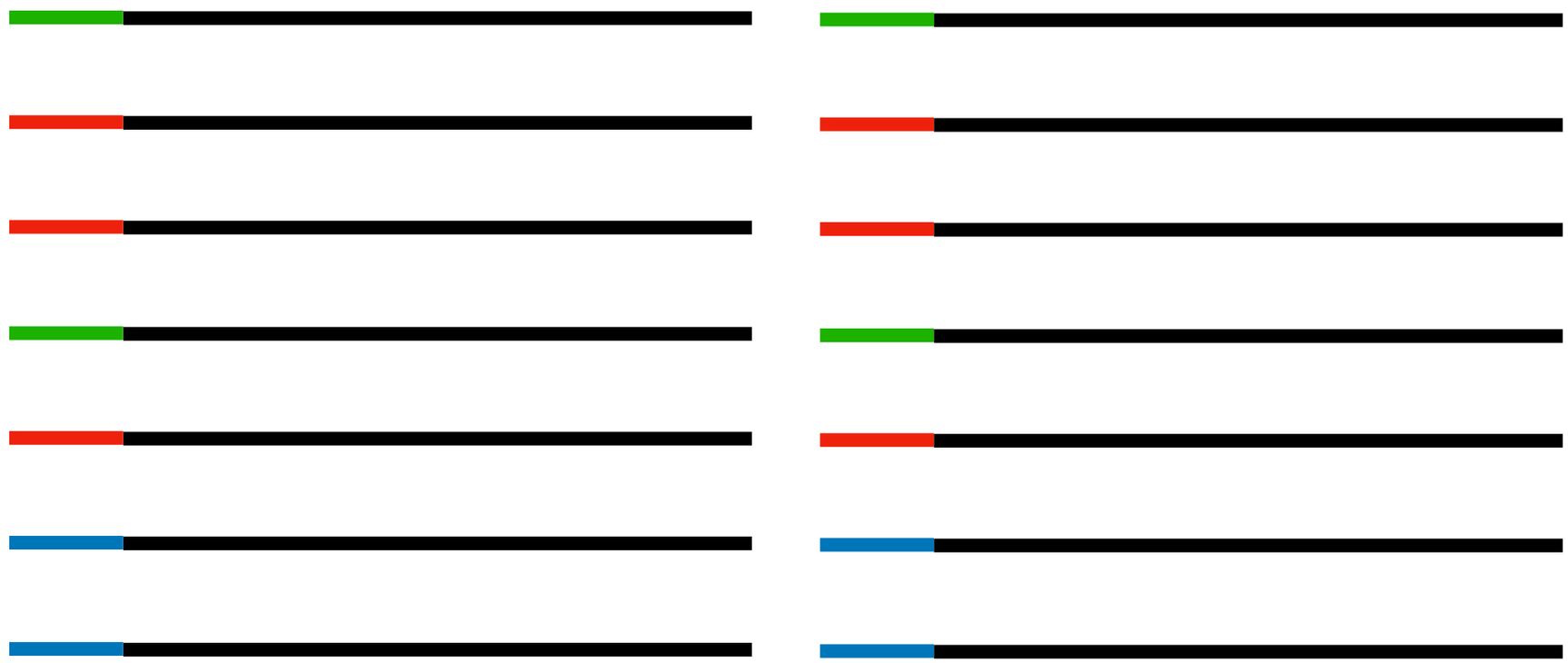
We need to demultiplex reads according to them, then cut them out

assign reads to its sample

How to demultiplex

Reads before demultiplexing

Paired Reads



Table

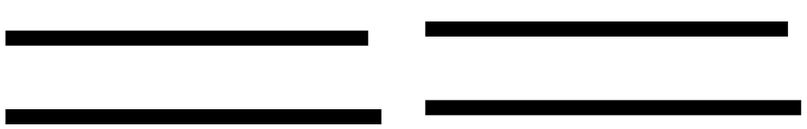
Index “green”: sample 1

Index “red”: sample 2

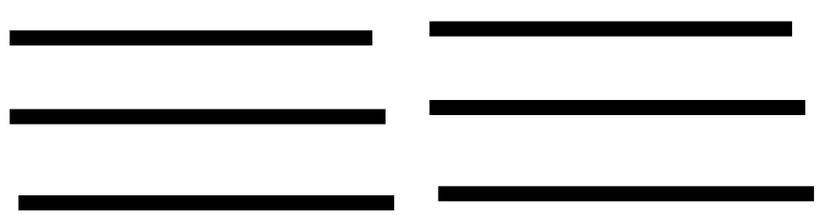
Index “blue”: sample 3

Reads after demultiplexing

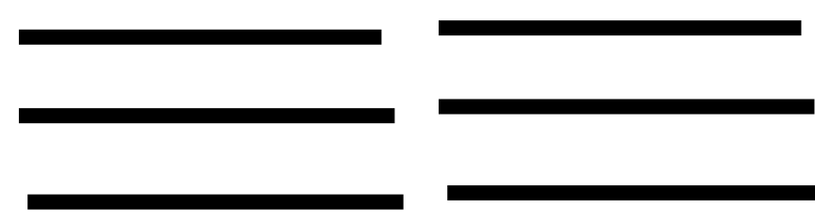
sample 1



sample 2



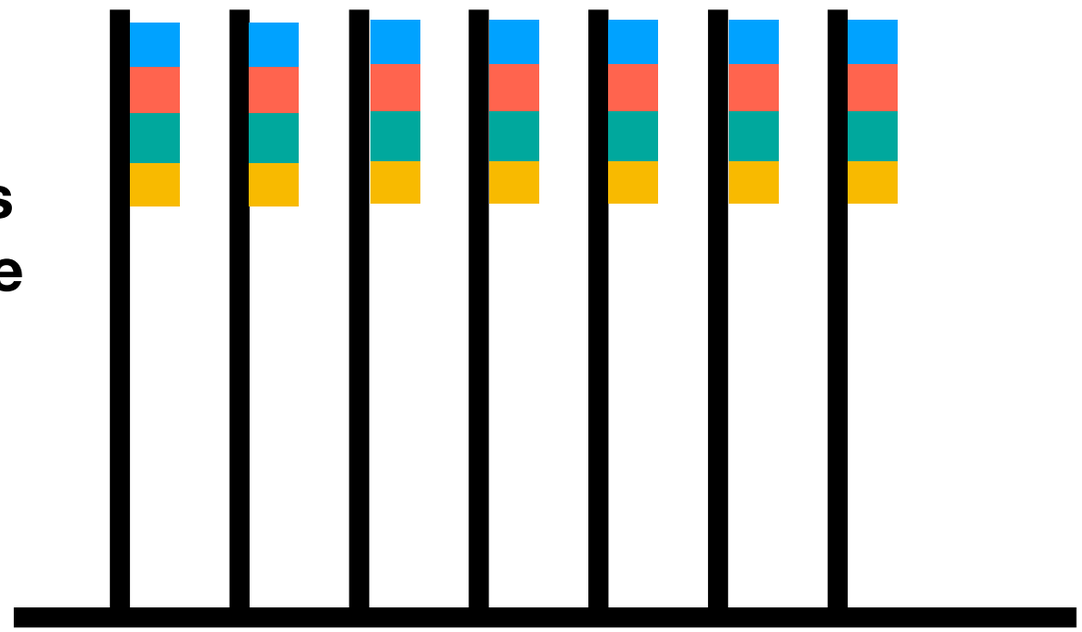
sample 3



Trim low quality bases and adaptor

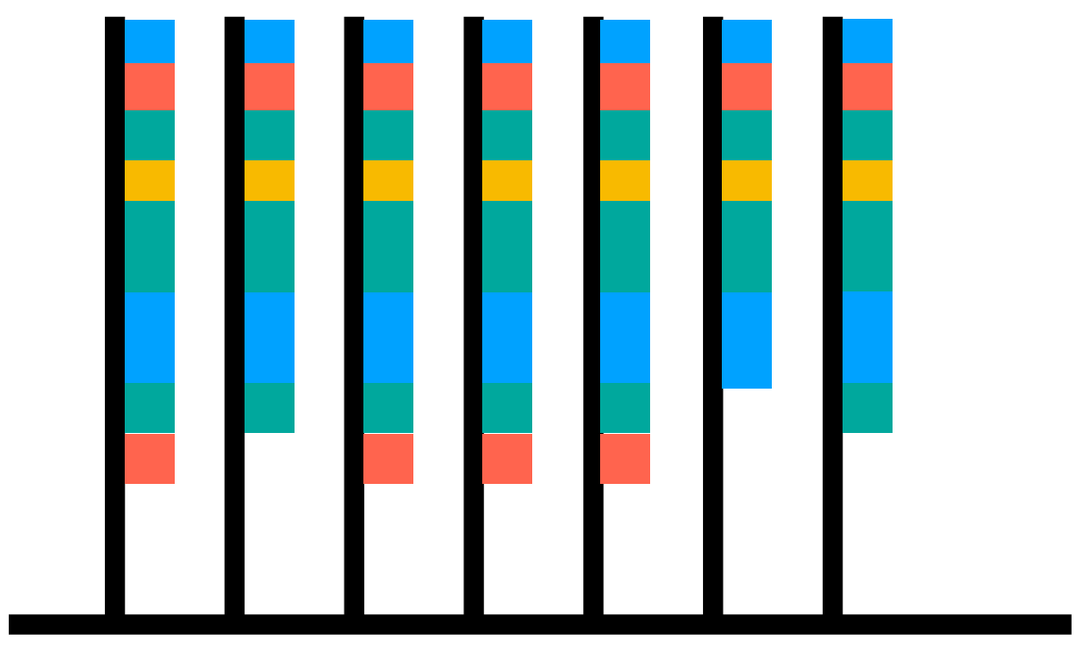
Why we need to trim low quality bases

A cluster of reads amplified from the same molecular



Last base is

Several rounds of extension later

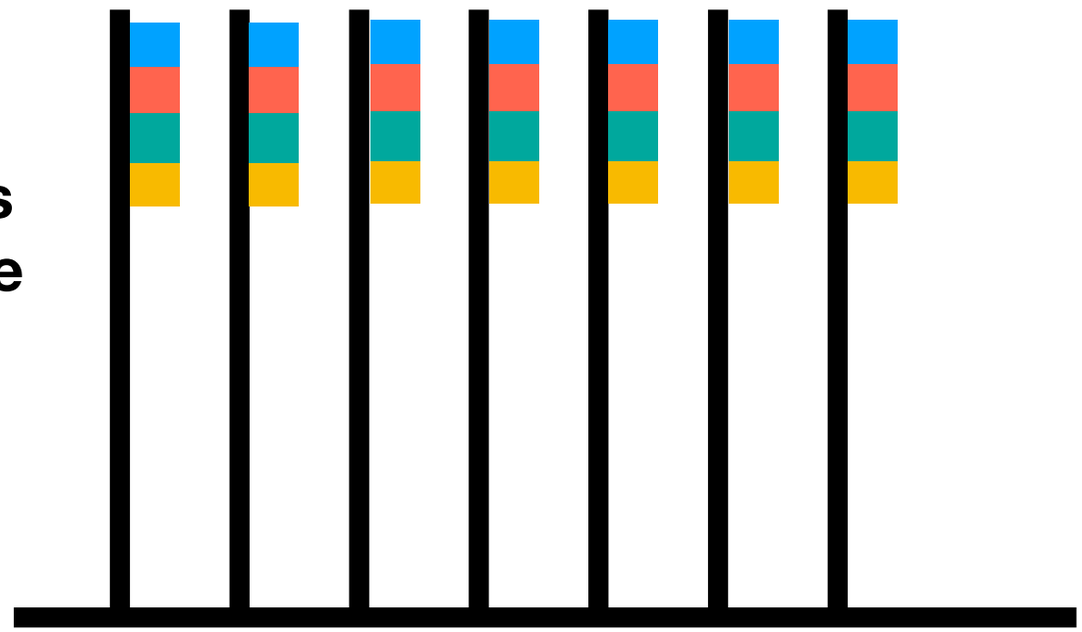


Last base is or ?

Trim low quality bases and adaptor

Why we need to trim low quality bases

A cluster of reads amplified from the same molecular

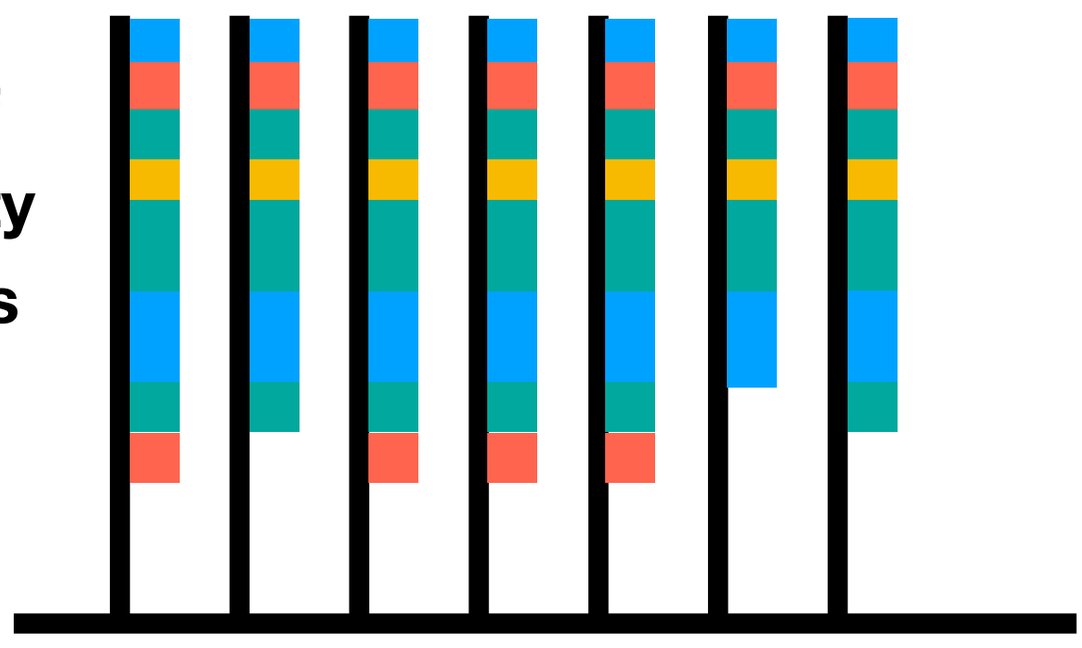


Last base is

High quality bases

Several rounds of extension later

longer the time of sequencing quality of bases becomes lower



Last base is or ?

Low quality bases

How to trim low quality bases

E(quality)>=15

ACGGCGTAGGCTGATGATCGGGTACGTCCGATCGTAGCTGTCA

E(quality)=30 GOOD

ACGGCGTAGGCTGATGATCGGGTACGTCCGATCGTAGCTGTCA

E(quality)=29 GOOD

ACGGCGTAGGCTGATGATCGGGTACGTCCGATCGTAGCTGTCA

E(quality)=25 GOOD

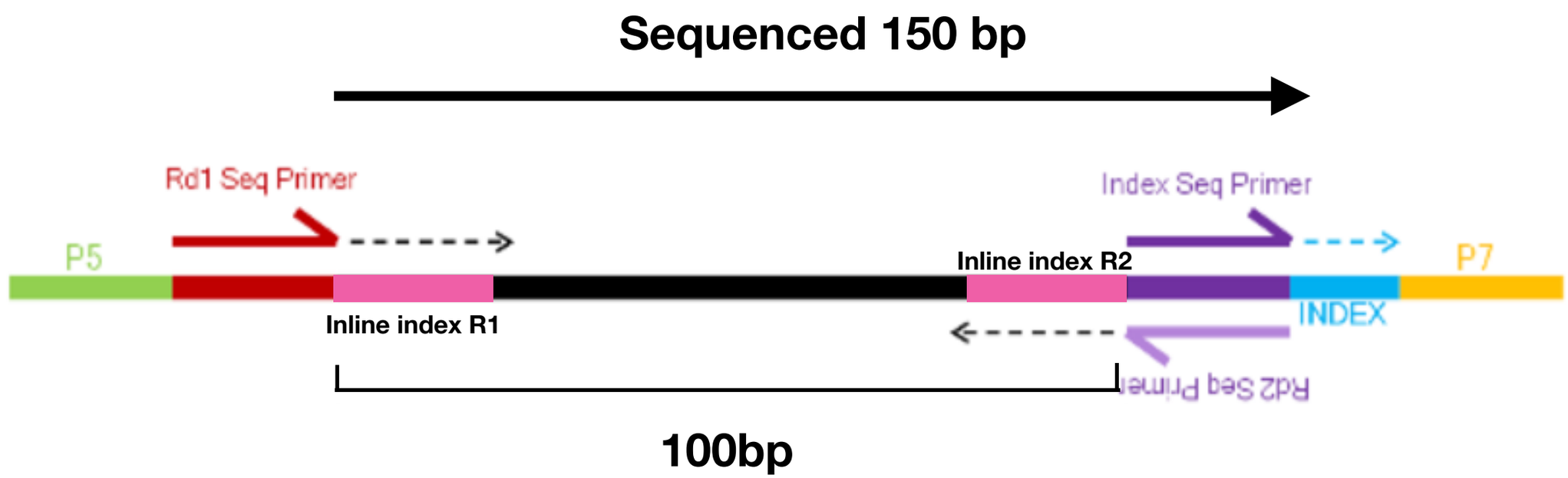
...

ACGGCGTAGGCTGATGATCGGGTACGTCCGATCGTAGCTGTCA

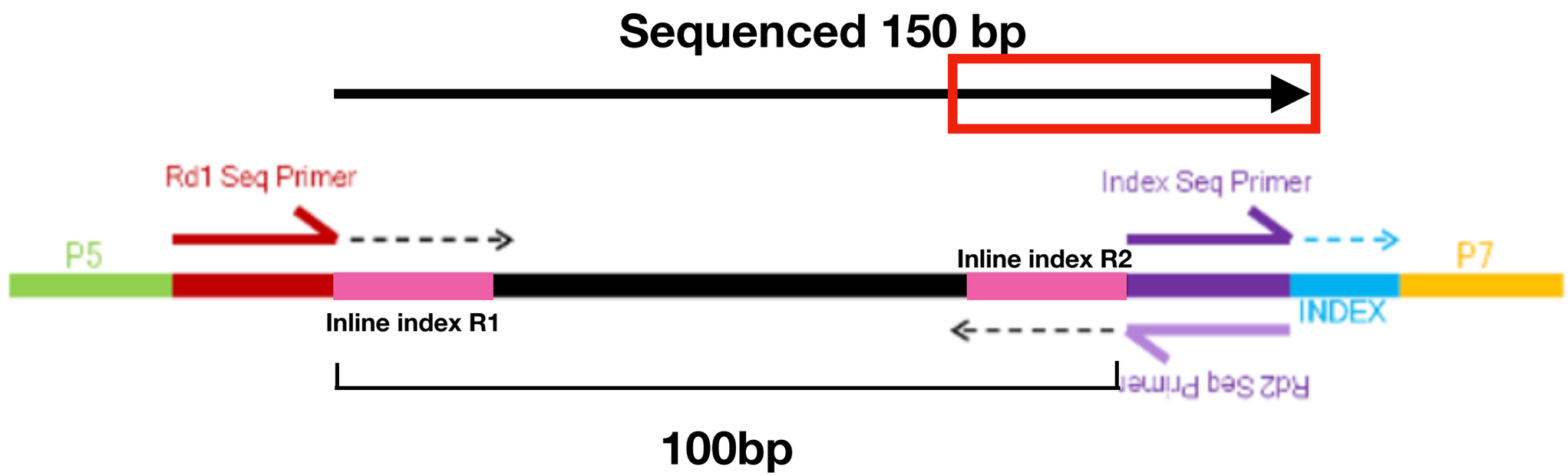
E(quality)=10 BAD!!

ACGGCGTAGGCTGATGATCG

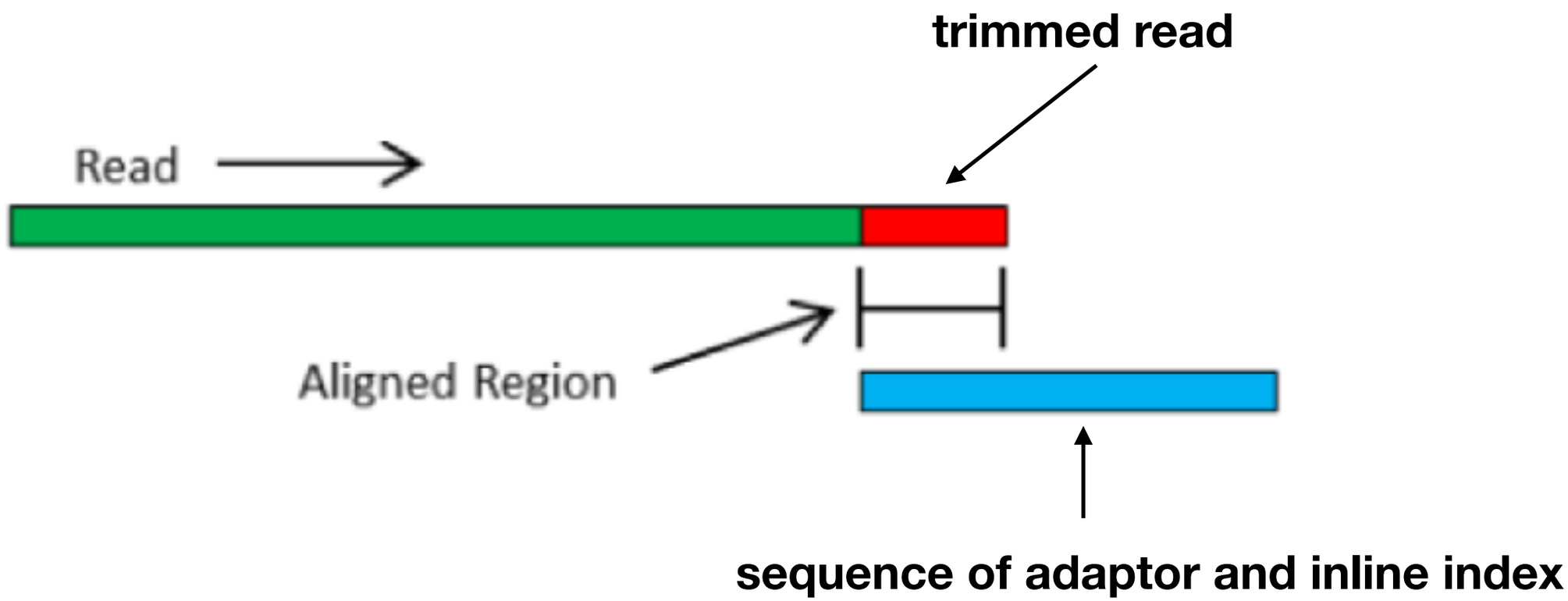
Why we need to trim adaptor?



Why we need to trim adaptor?



How to trim adaptor?



Reads have been cleaned. Let’s start assemble !

What’s assemble

Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

Reads 2: TCTGATTCGGTTCGGATCTGGGCAT

Reads 3: ATCTGGGCATGGCGTTCGATGTCGCTAT

3 reads in a sample

What’s assemble

Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

Reads 2: TCTGATTCGGTTCGGATCTGGGCAT

Reads 3: ATCTGGGCATGGCGTTCGATGTCGCTAT

Resulting contig:

Contig1 CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCAT

“Contig” is the sequence assembled from the reads

What’s assemble

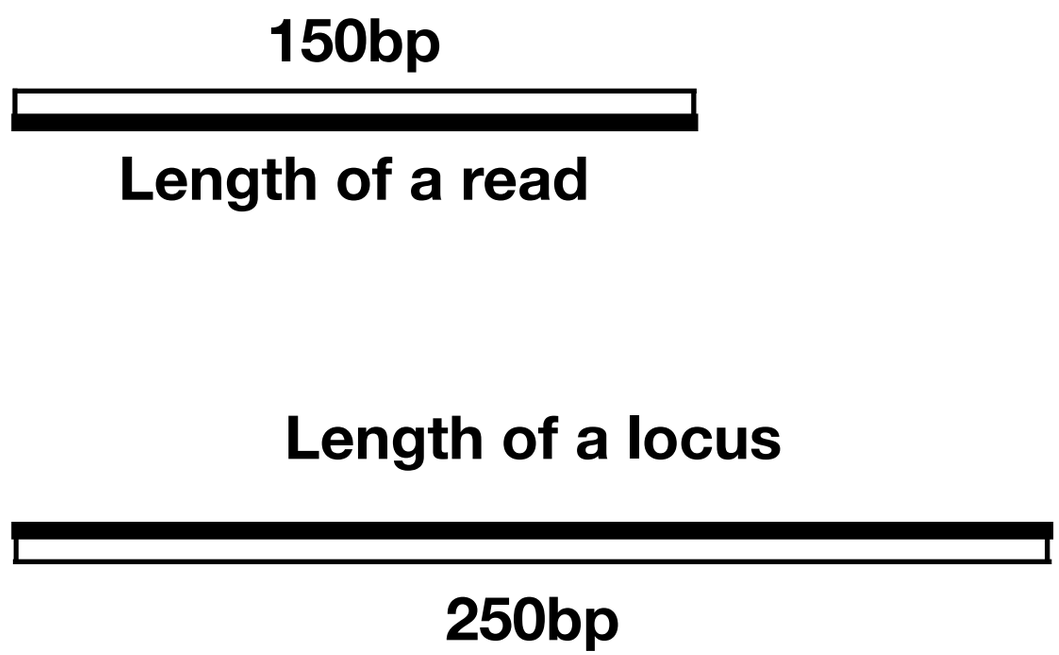
Contig1 CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCAT

Reads 3: ATCTGGGCATGGCGTTCGATGTCGCTAT

Resulting contig:

CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCATGGCGTTCGATGTCGCTAT

Why raw data need to be assembled before various analysis?



Reads are too short to reach the length of the locus

How raw reads magically become sequences of loci of each sample?

Remove PCR duplicates

Parse reads to loci

Assemble parsed reads

Further assemble

Get orthologue assemblies

Remove PCR duplicates

Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

PCR duplicate of Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

PCR duplicate of Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

Reads 2: TCTGATTCGGTTCGGATCTGGGCAT

Resulting contig:

Contig1 CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCAT

Remove PCR duplicates

Reads 1: CGGCGGATCTGATGGGATCTGATTCGGTT

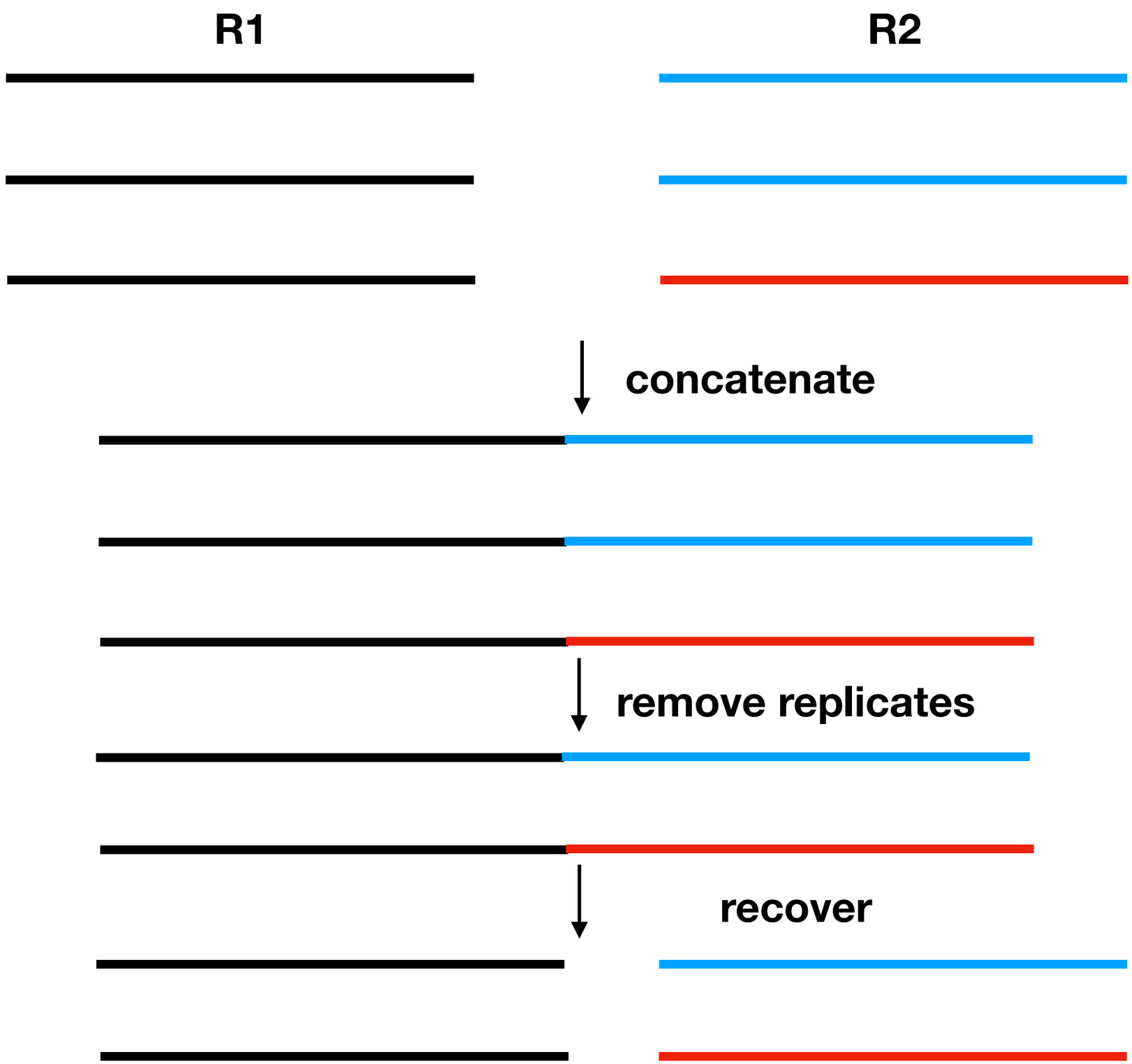
Reads 2: TCTGATTCGGTTCGGATCTGGGCAT

Resulting contig:

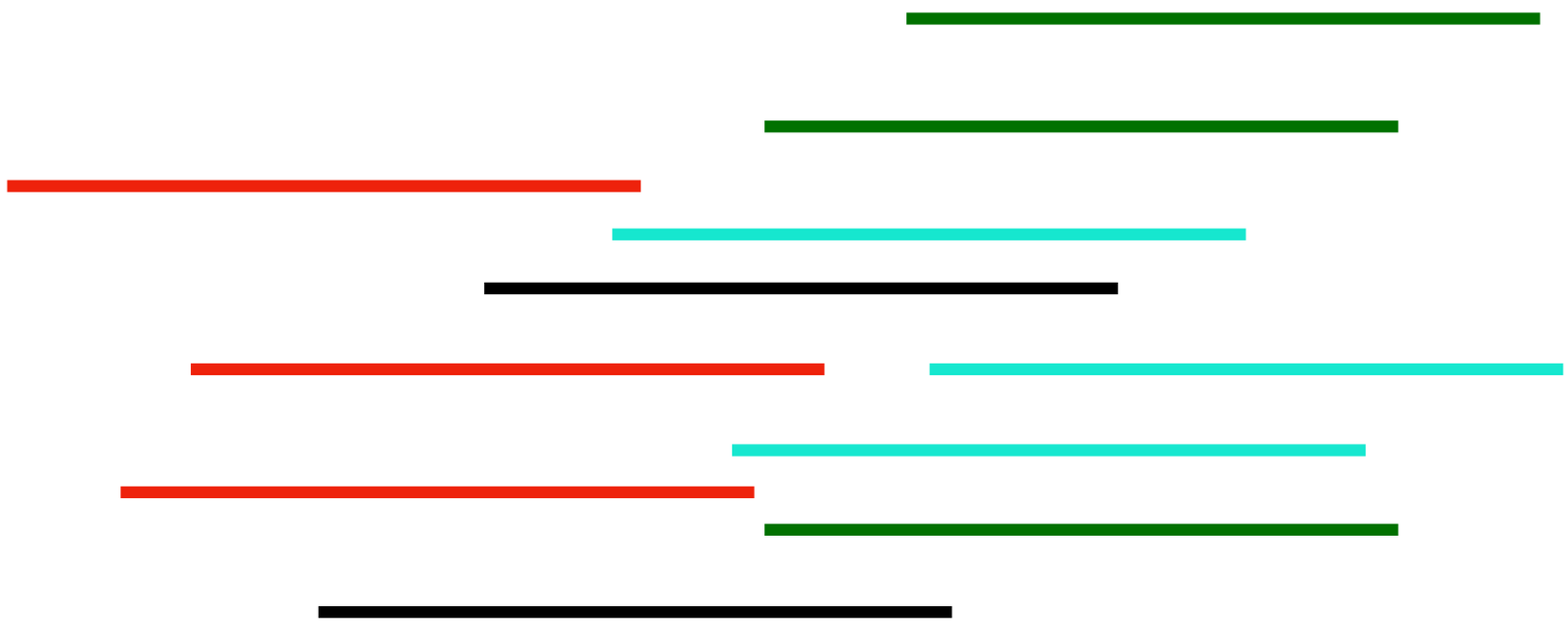
Contig1 CGGCGGATCTGATGGGATCTGATTCGGTTCGGATCTGGGCAT

PCR duplicates are redundant for assembly

Remove PCR duplicates



Parse reads to loci

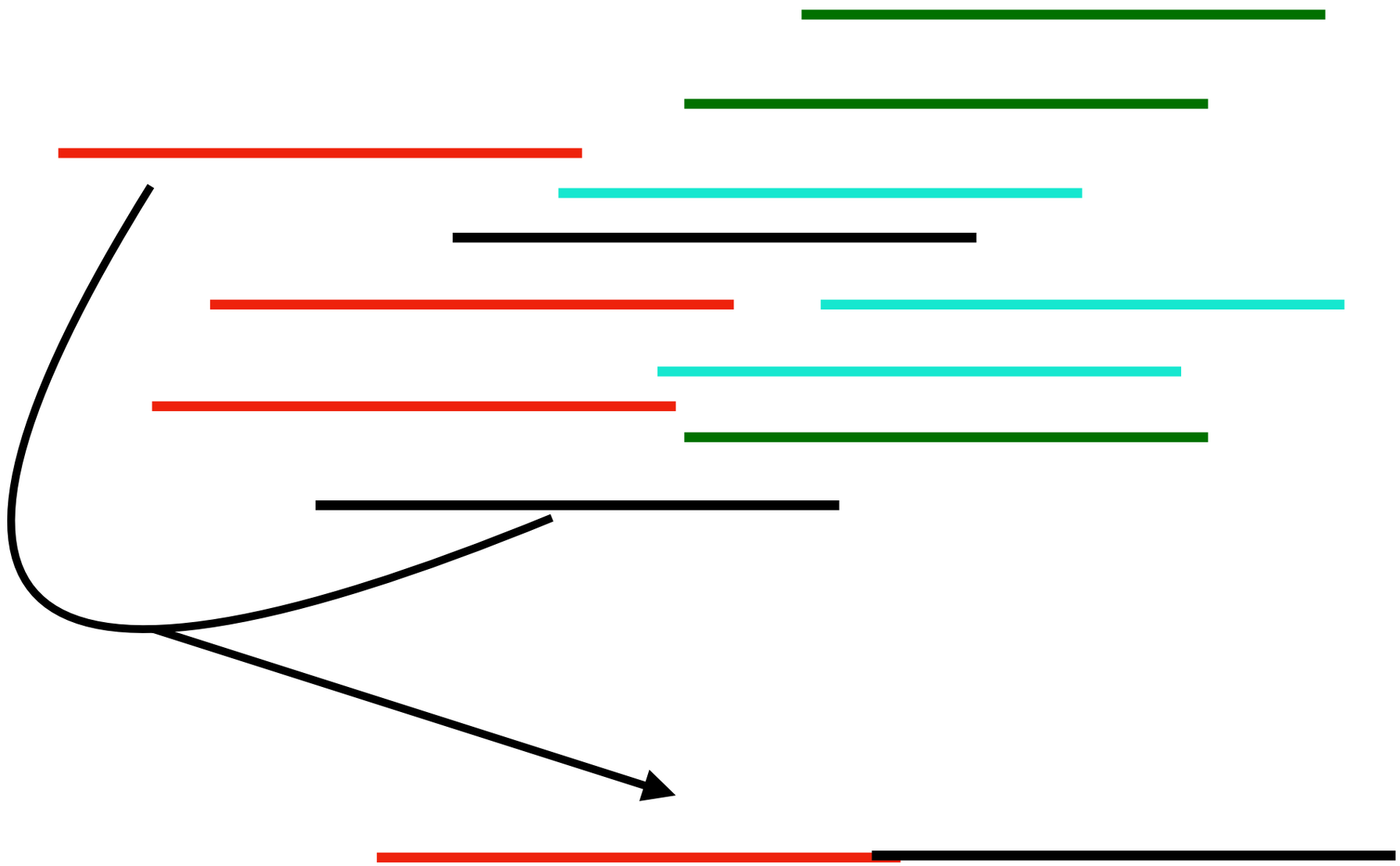


Mixed short reads from lots of loci

Reads of the same color indicate they come from the same loci

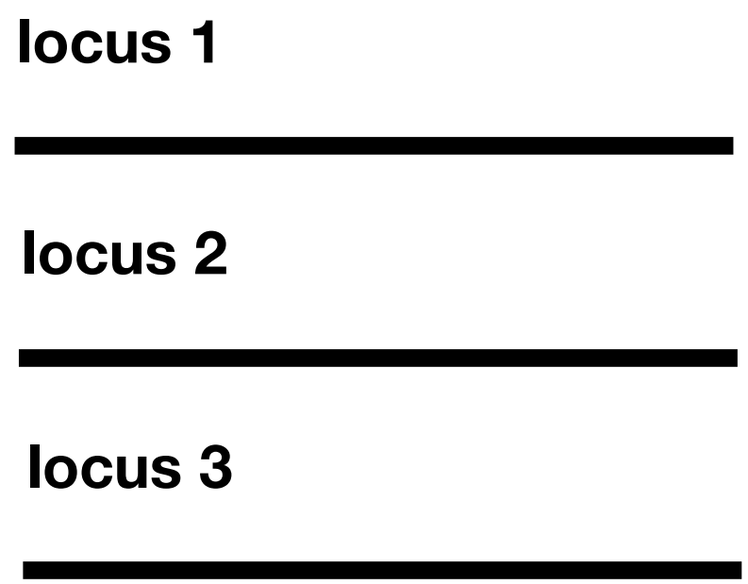
Parse reads to loci

Parse reads to loci



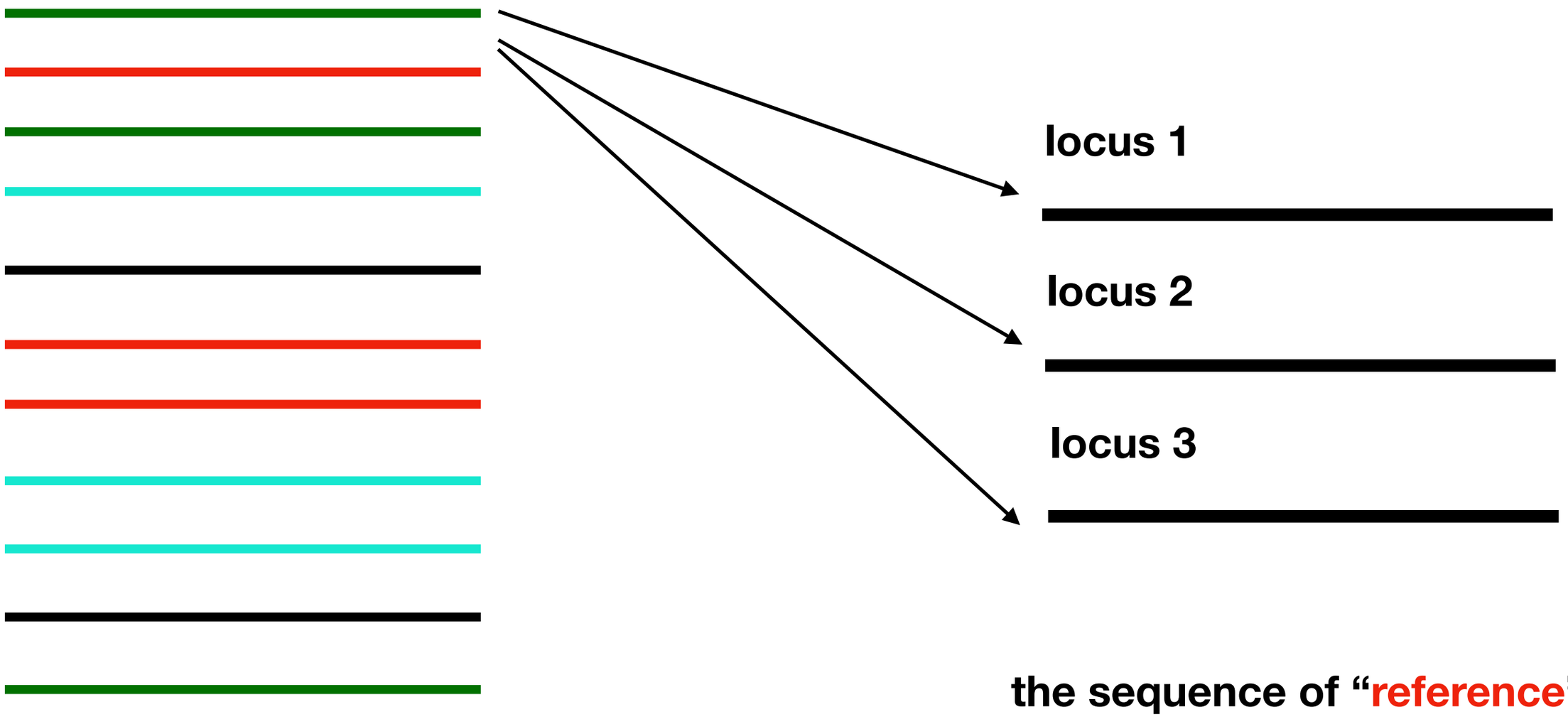
If reads from different loci assembled together, the resulting contig will be "chimera"

Parse reads to loci



Remember me? I'm the sequence of "reference", used to design the baits

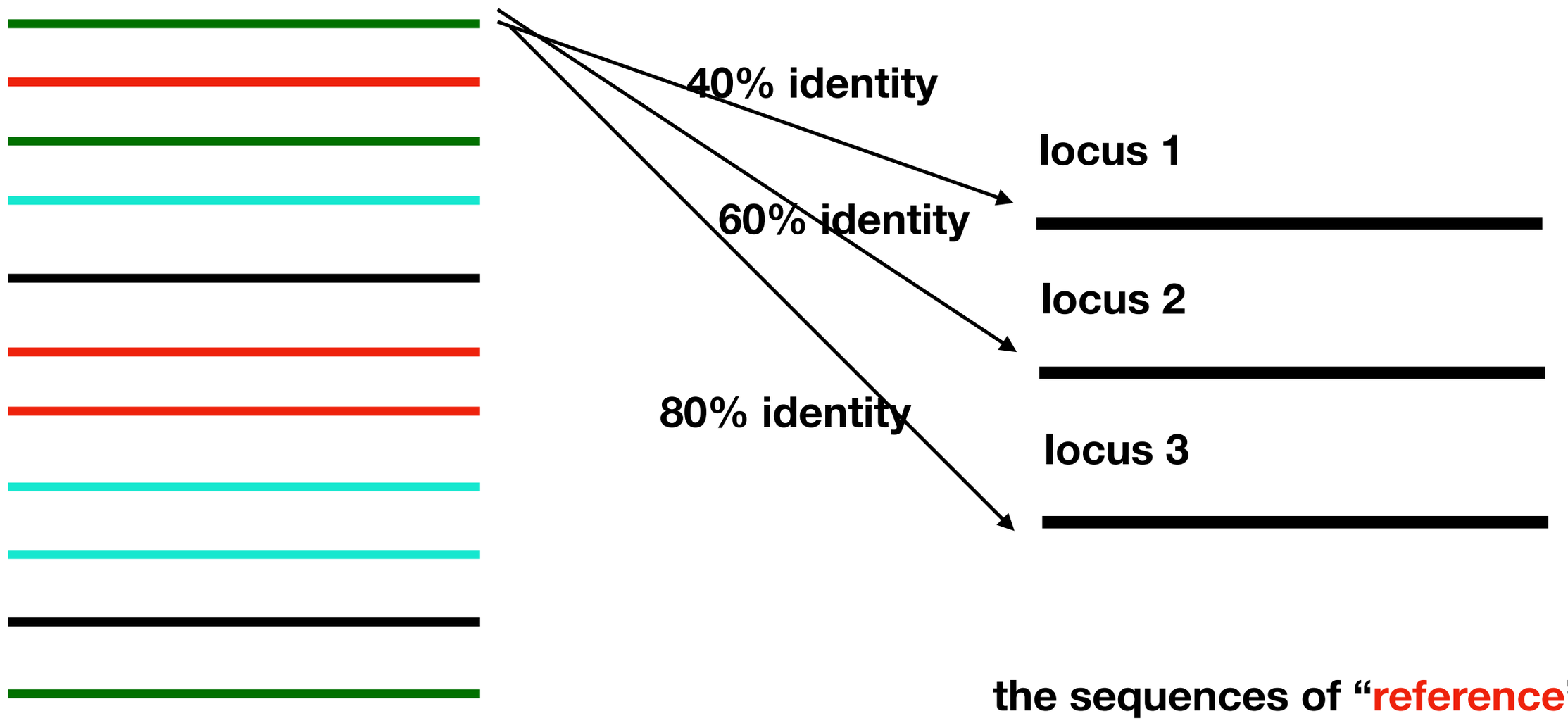
Parse reads to loci



Mixed short reads from several loci

Compare reads with each locus

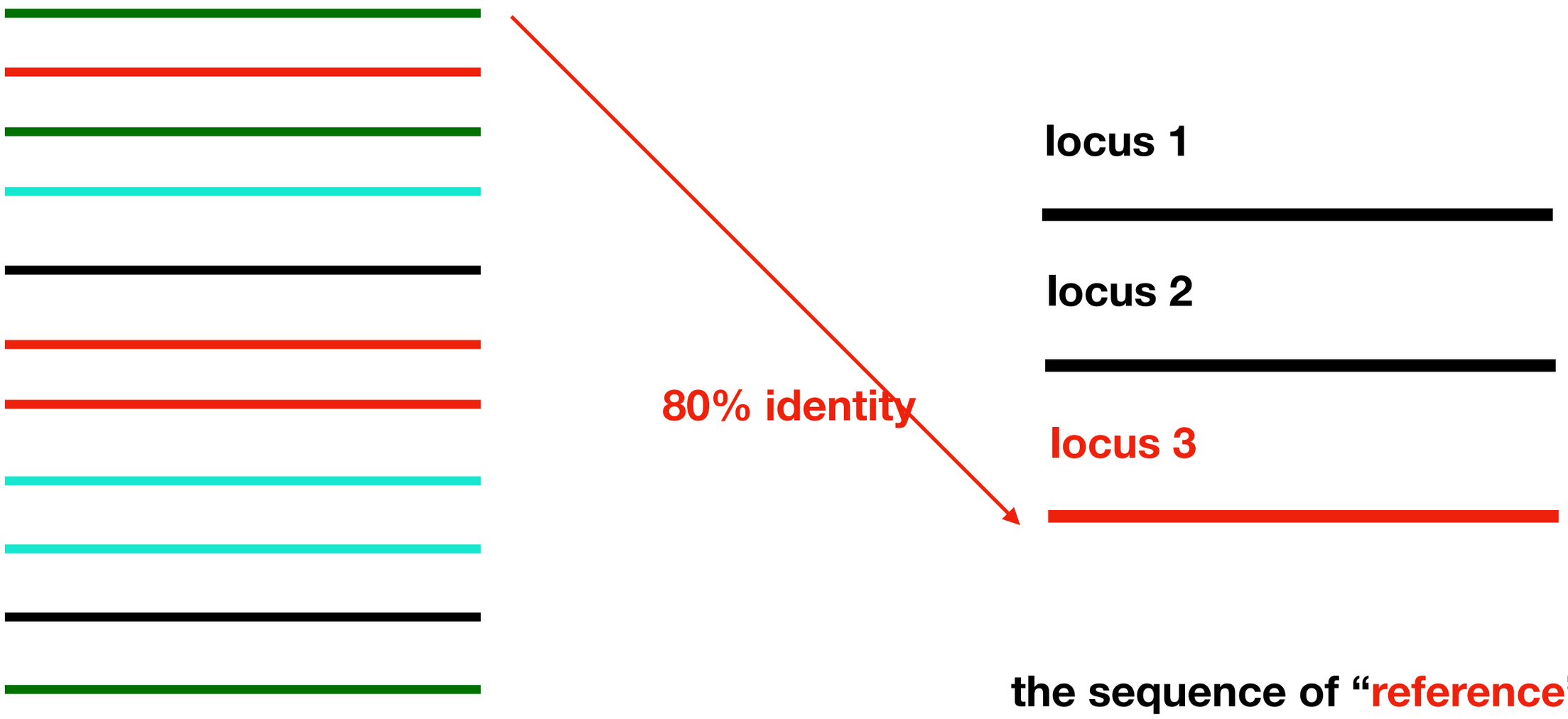
Parse reads to loci



Mixed short reads from several loci

Reads got different identity with each locus

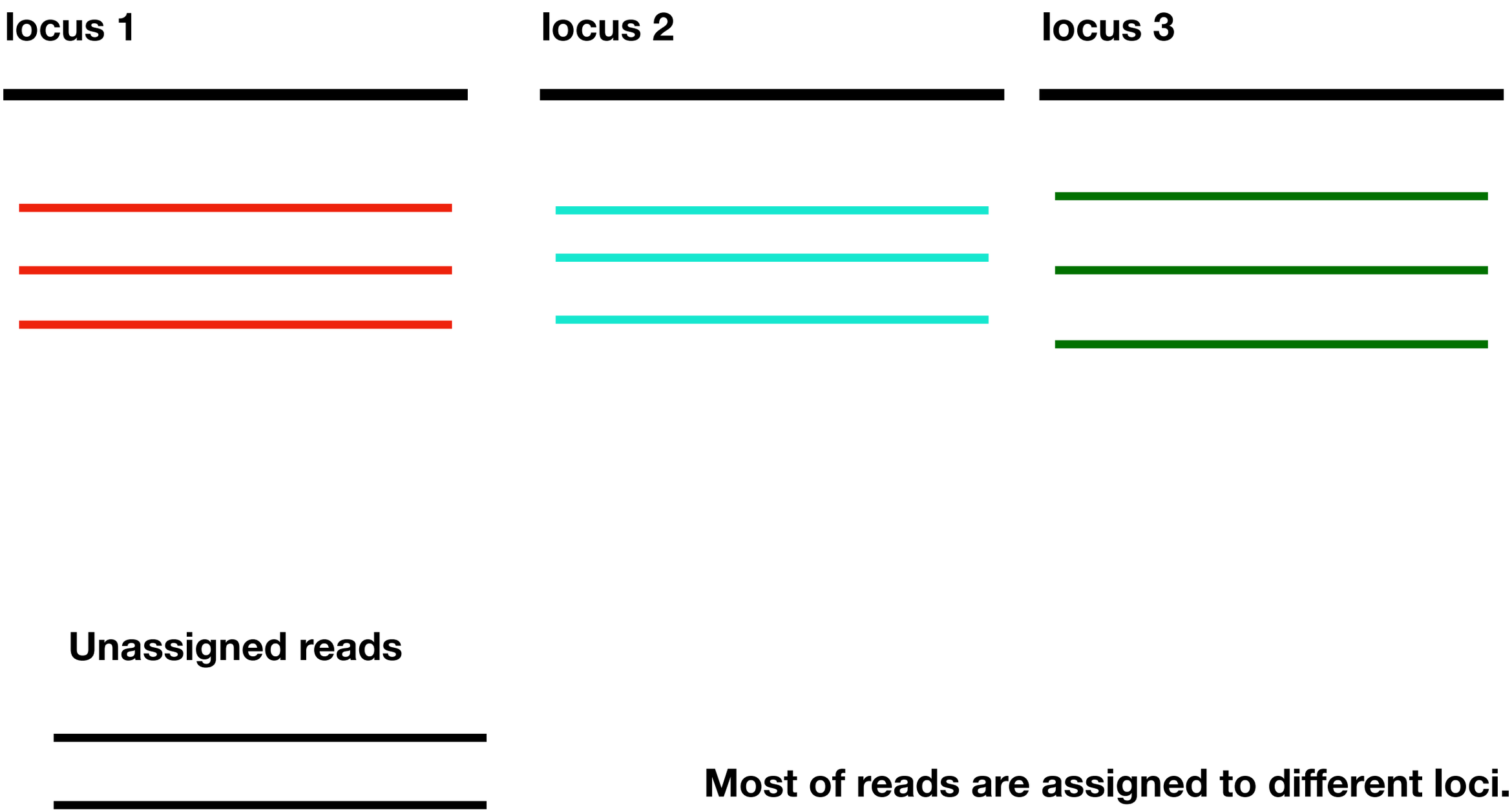
Parse reads to loci



Mixed short reads from several loci

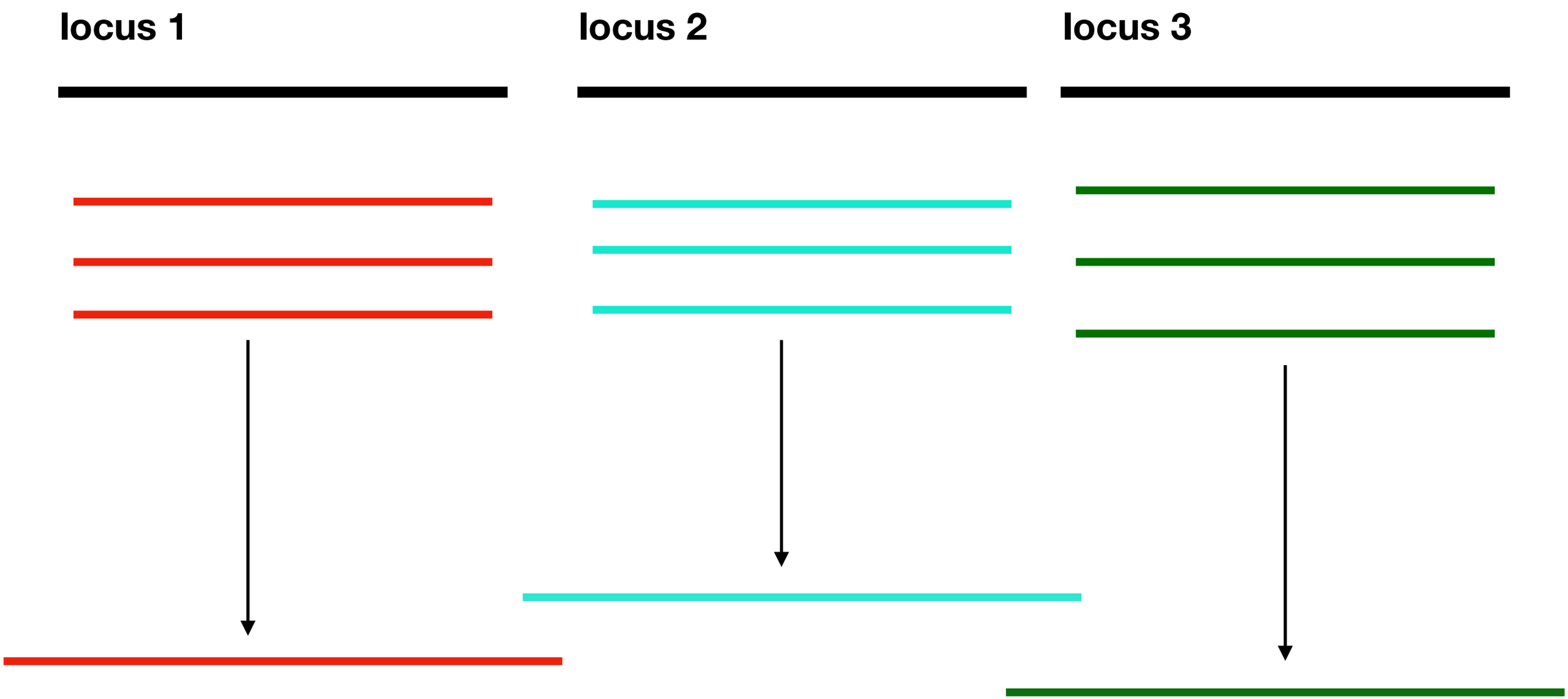
Select reads with highest identity

Parse reads to loci



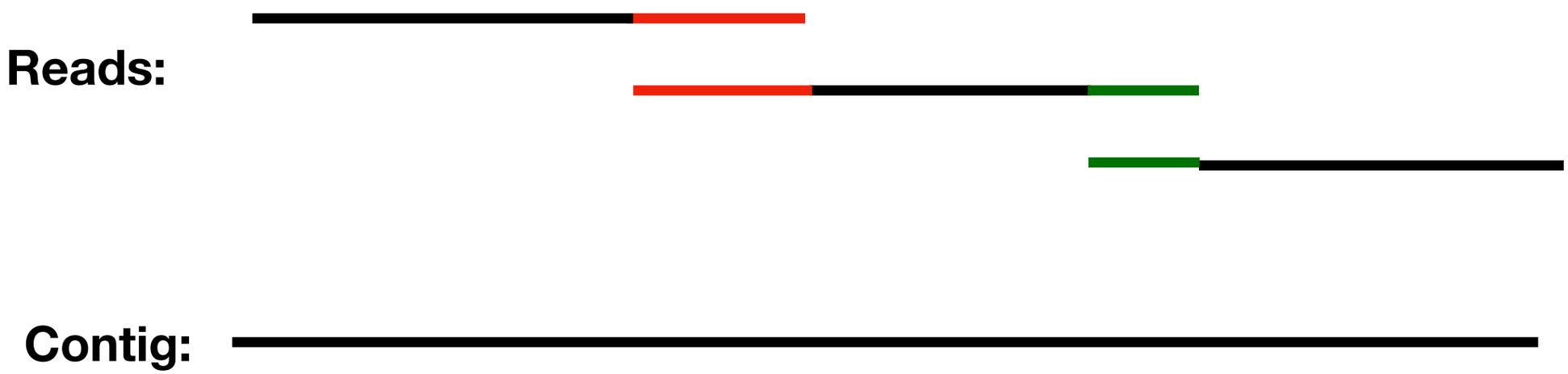
Most of reads are assigned to different loci.
Some reads from nowhere are still unassigned

Assemble parsed reads

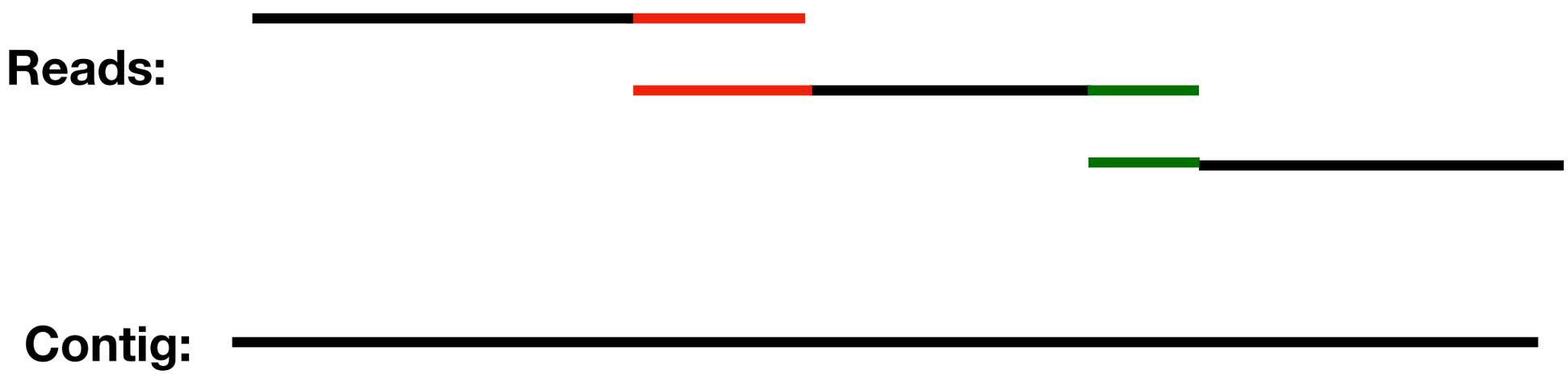


Assemble parsed reads into longer contigs

Assemble parsed reads



Assemble parsed reads



In real case, question is not that easy. We always have loci assigned with more than 2,000 reads

Assemble parsed reads

Find overlaps among all reads

Build a graph recording overlaps among all reads

Traverse through the graph to get contigs

Assemble parsed reads

Find overlaps among all reads

Align the reads

K-mer

FM-index

Assemble parsed reads

Find overlaps among all reads

Align the reads

K-mer

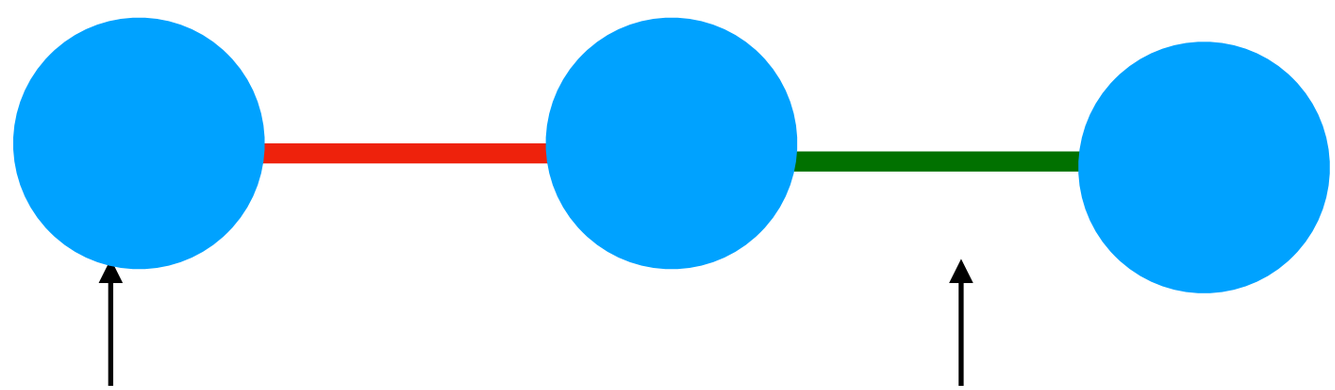
FM-index

Only considerable length of overlap between reads will be kept (25 bp), to guarantee the low probability of accidentally overlap occurs

Assemble parsed reads

Build a graph recording overlaps among all reads

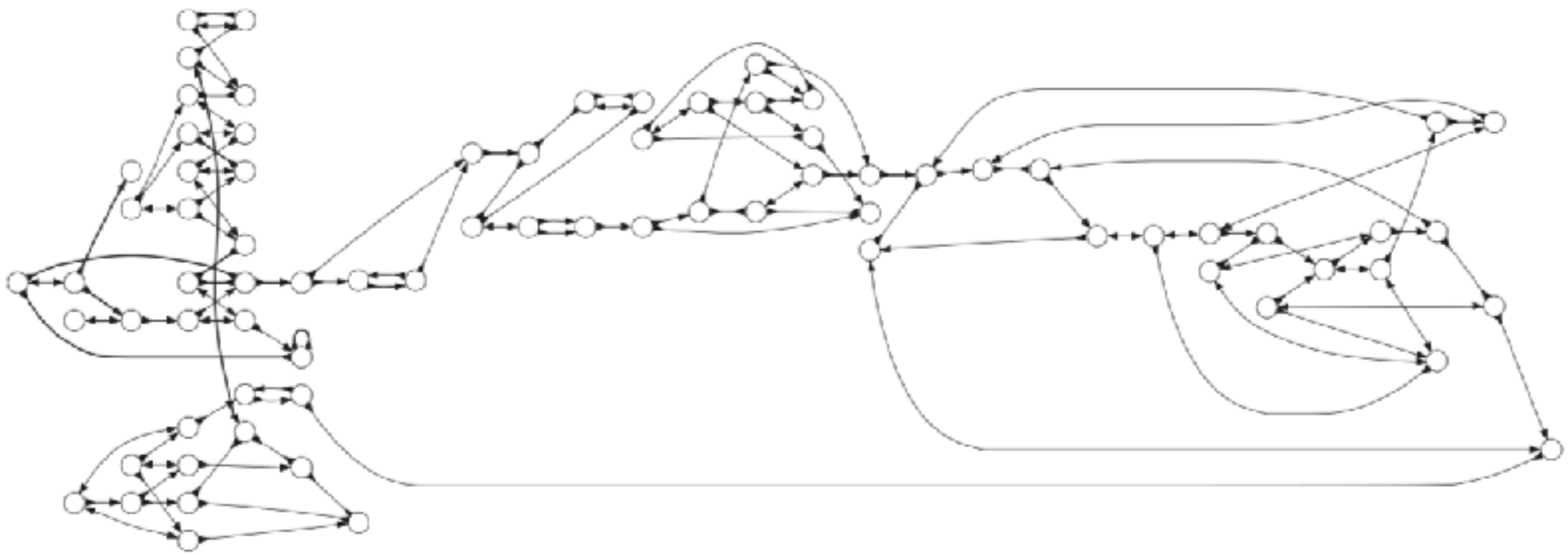
Reads:



Circle represent the read we call it "node" or "vertex"

Line represents the connection between reads called "edge"

Assemble parsed reads



Graph of *Campylobacter jejuni*

Assemble parsed reads

Traverse through the graph to get contigs

For each locus, there's only one sequence

Traverse through the all nodes in the graph and each node only pass once

Assemble parsed reads

Traverse through the graph to get contigs

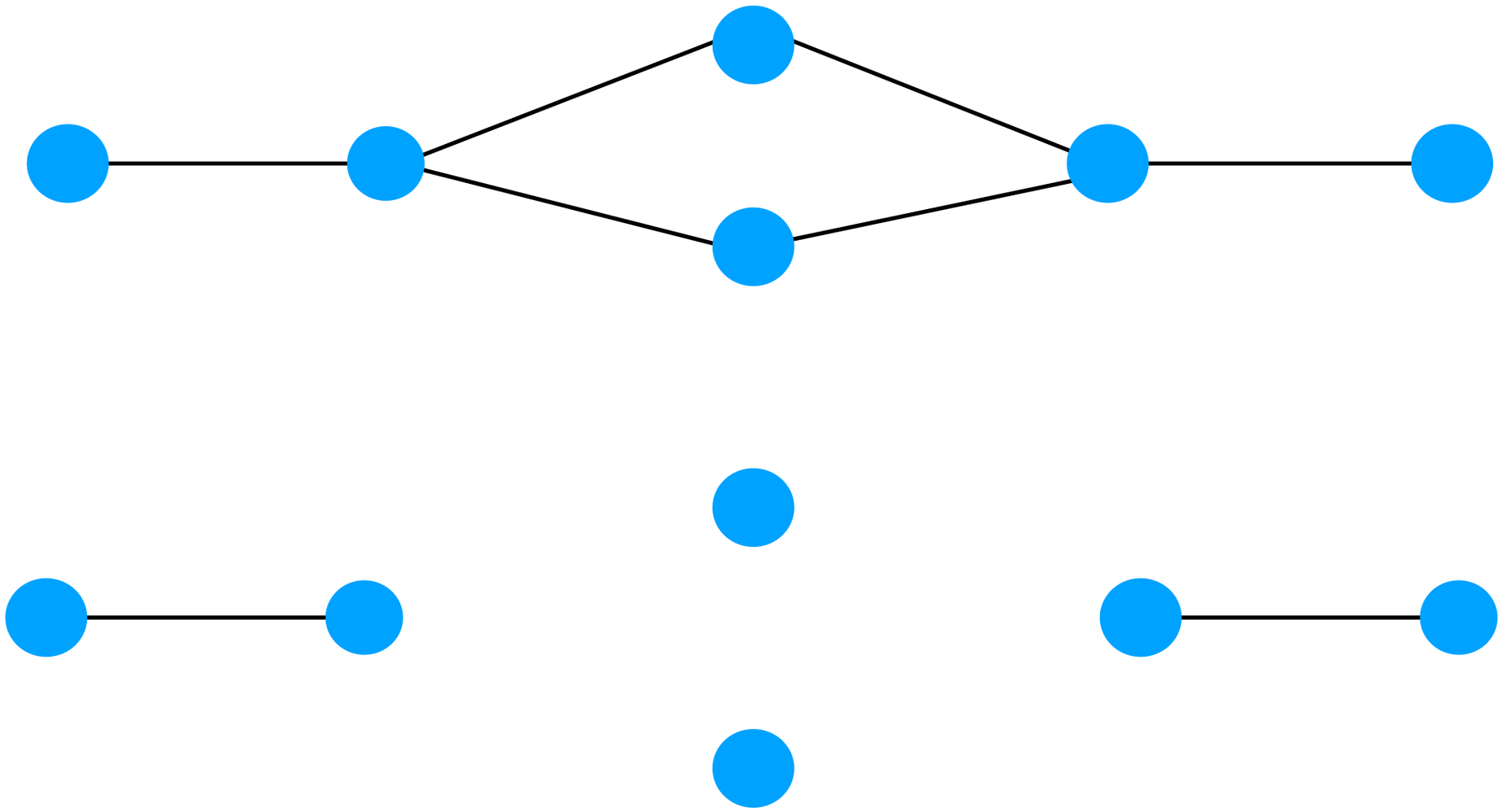
For each locus, there's only one sequence

Traverse through the all nodes in the graph and each node only pass once

But this assumption is hard to fulfill

Assemble parsed reads

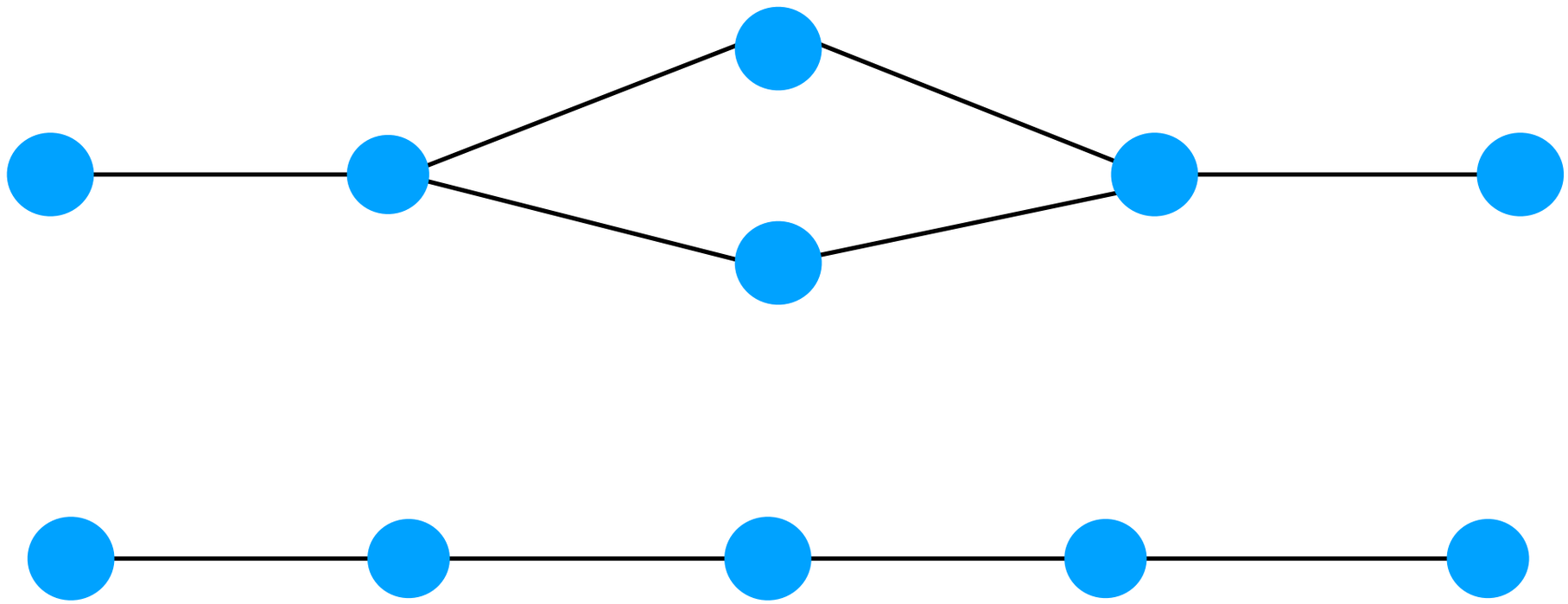
Bubble:



Break the graph into several sub-graph

Assemble parsed reads

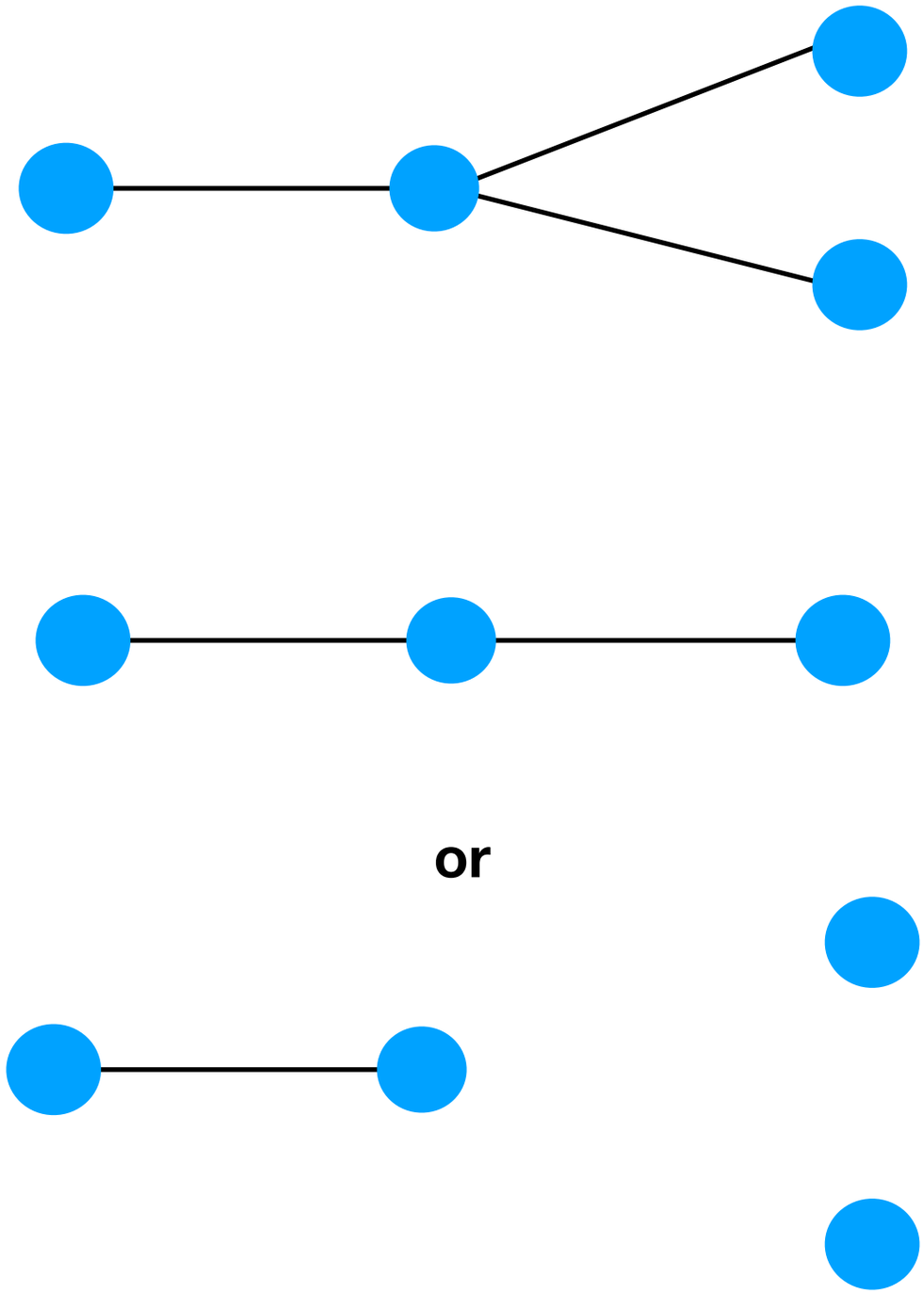
Bubble:



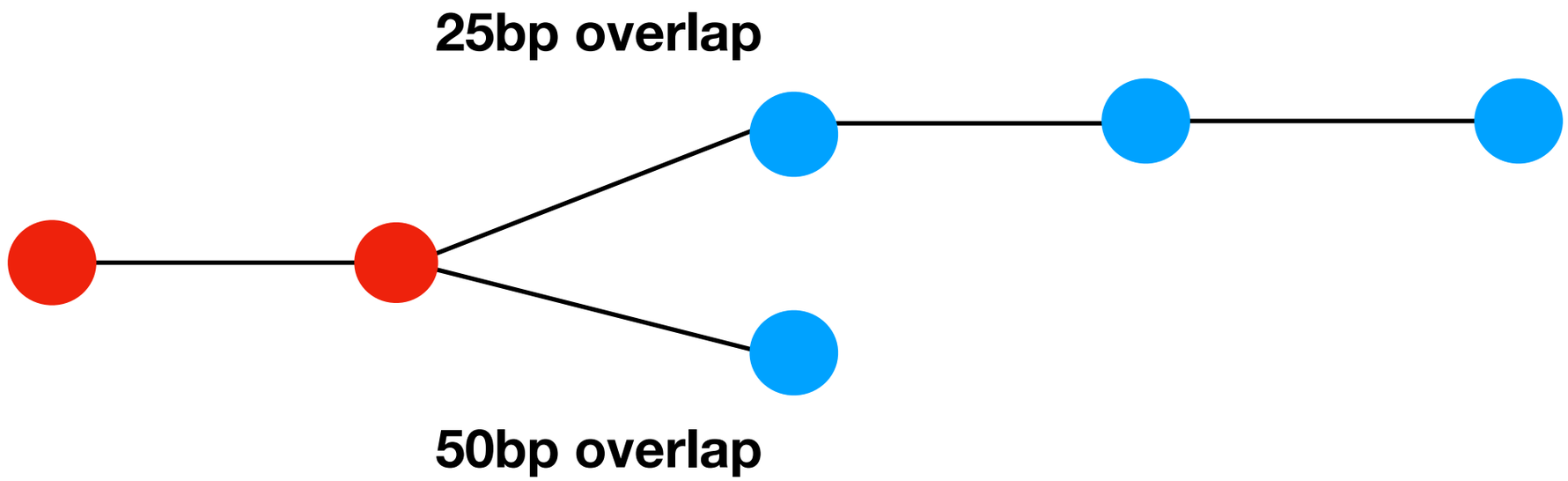
Discard one of the path

Assemble parsed reads

Tip:



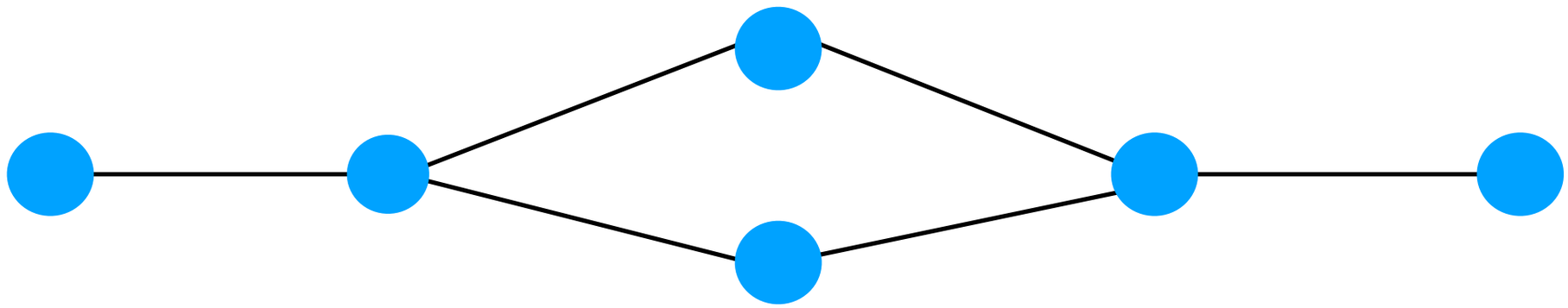
Why we need graph



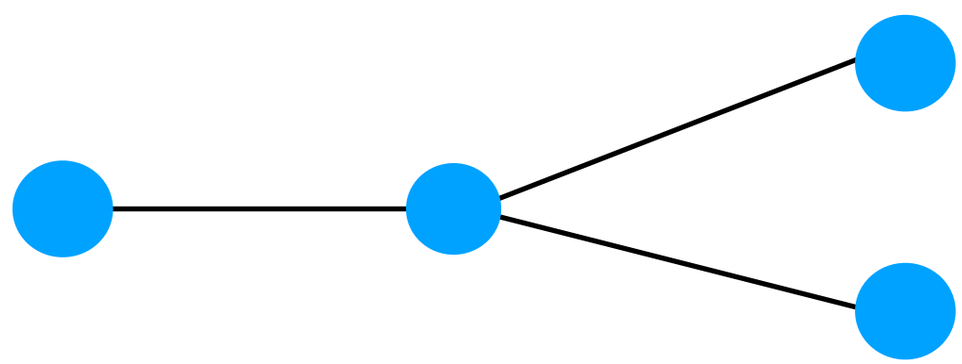
Which way you should choose

Further assemble

Bubble:



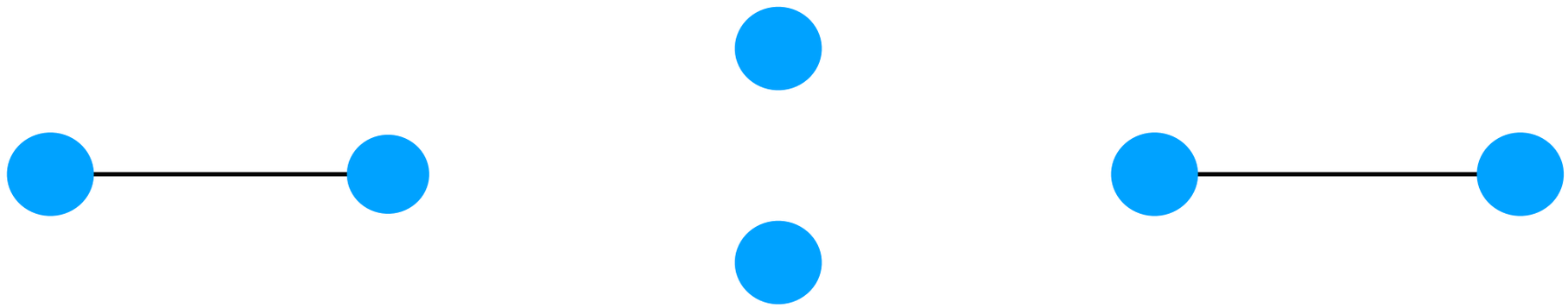
Tip:



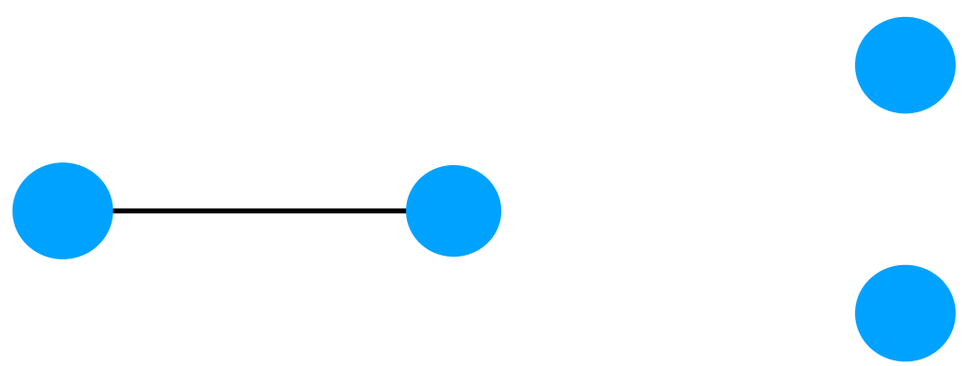
If two path is too diverged (>= 95% identity). The path will be split into several contigs

Further assemble

Bubble:

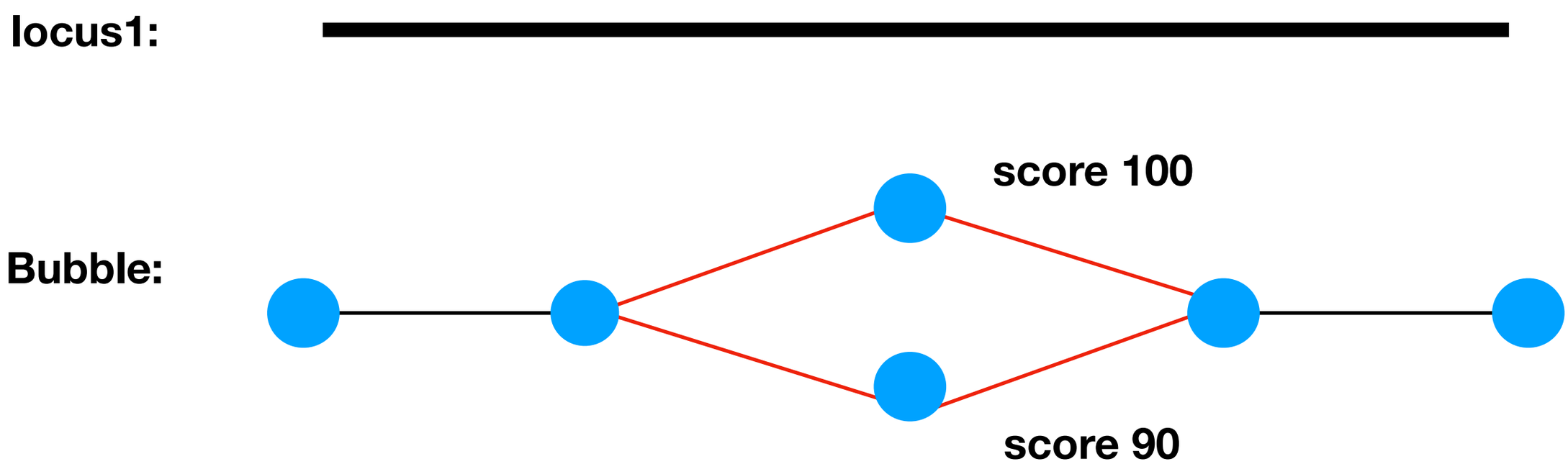


Tip:



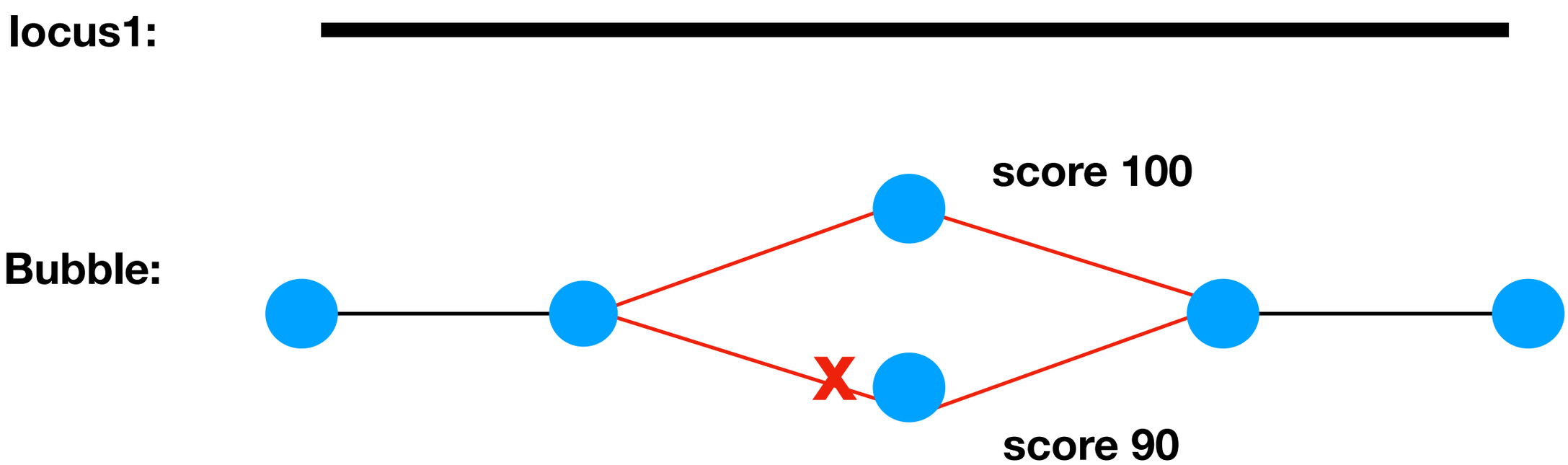
If two path is too diverged (>= 95% identity). The path will be split into several contigs

Further assemble



Each contig will be aligned to reference. Graph will be reconstructed. The alignment score of each path will be calculated.

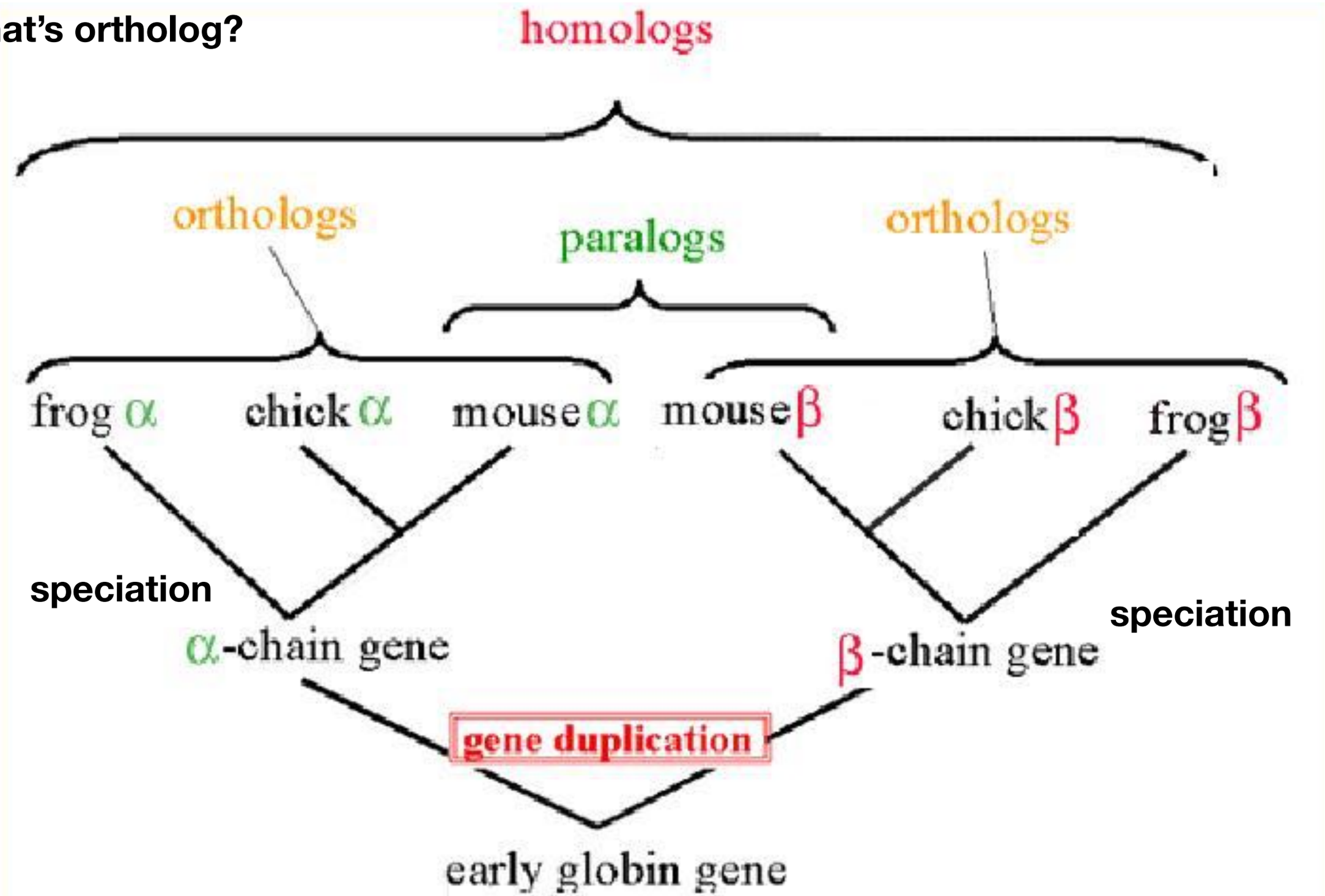
Further assemble



Keep the path with higher score.

Get orthologous assemblies

What's ortholog?



Get orthologous assemblies

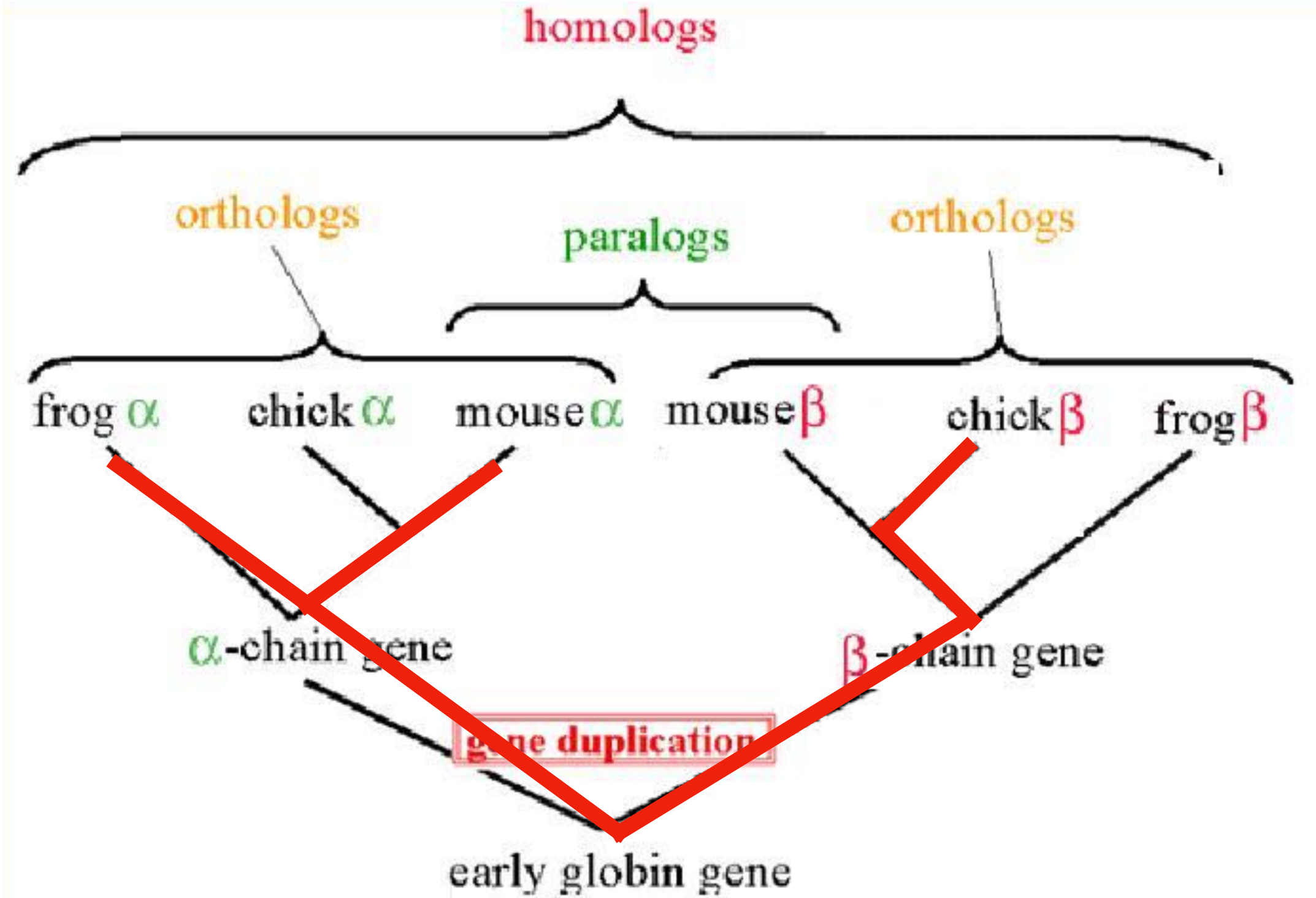
Why we need it?

Our final aim is to reveal evolutionary history among our enriched species

Orthologues genes are derived from "speciation event". So, the evolutionary history of these genes are identical with the evolutionary history of species

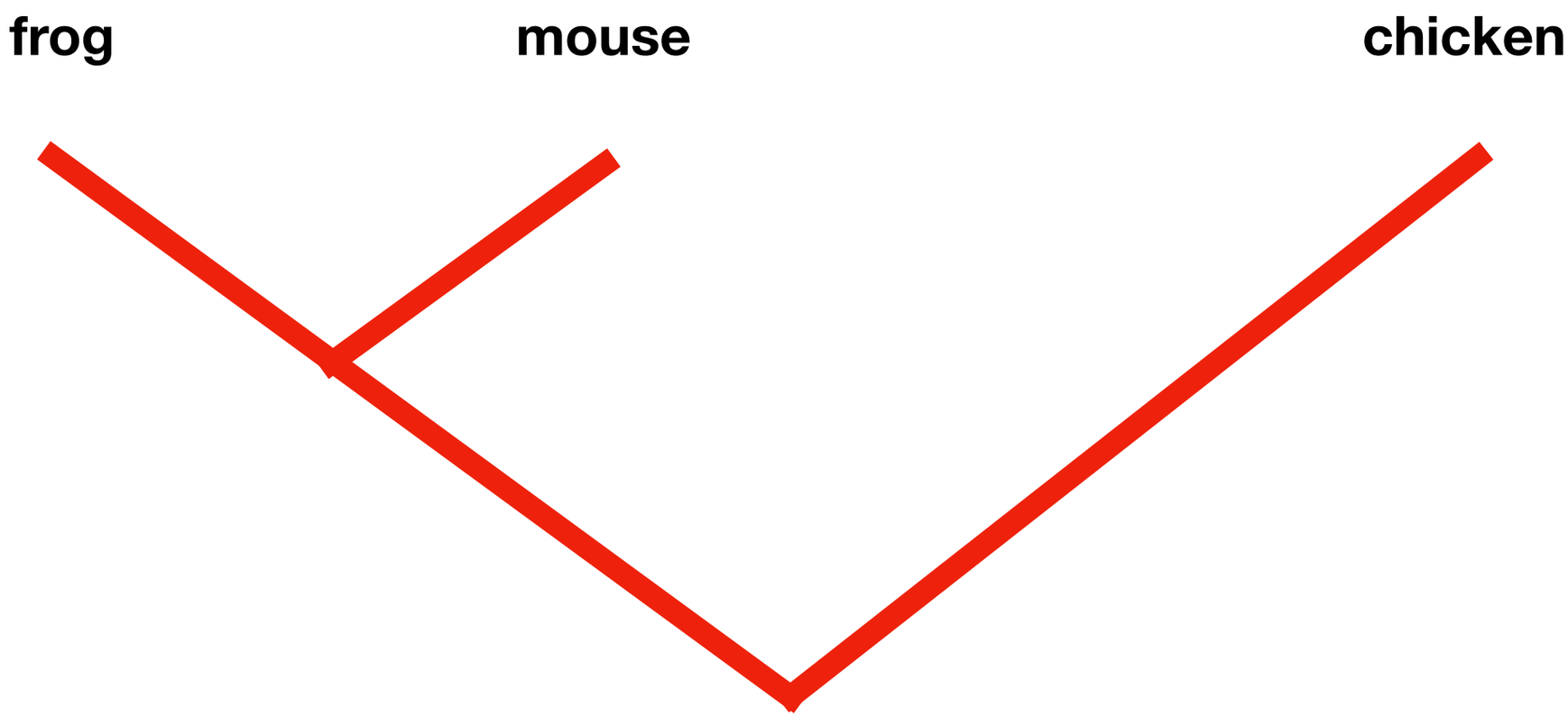
Get orthologous assemblies

What will happen if we use paralog genes to reveal evolutionary history



Get orthologous assemblies

What will happen if we use paralog genes to reveal evolutionary history



Get orthologous assemblies

There's various way to find orthologues. The method we used here called reciprocal blast

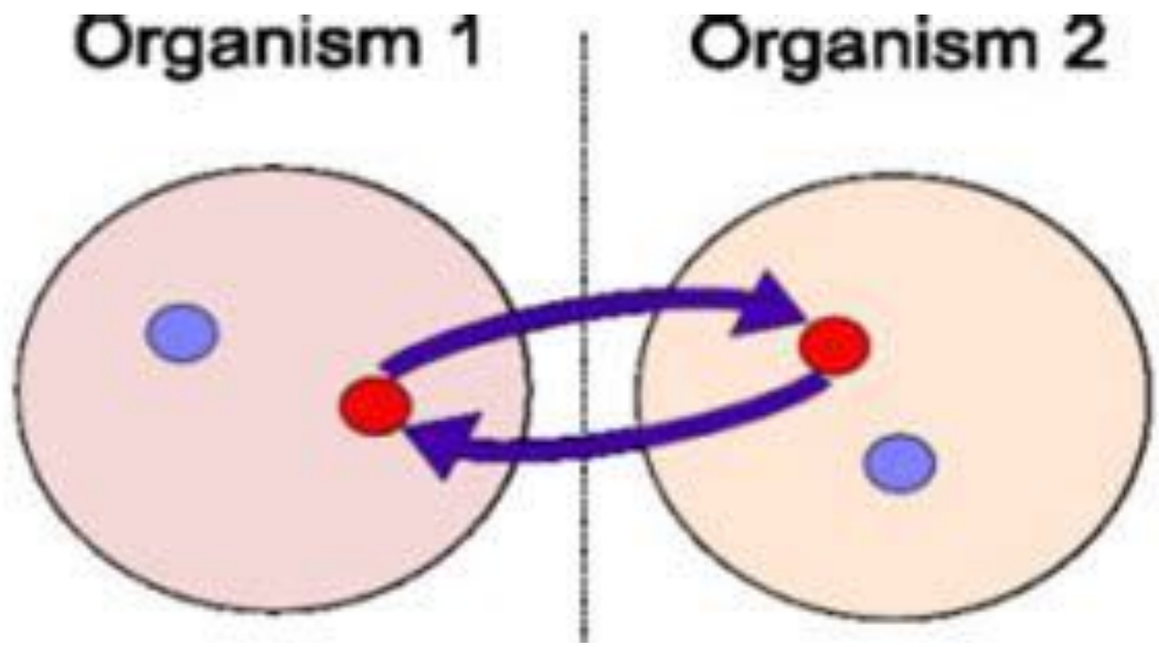
This method is built on the assumption that orthologous genes have identical or highly related functions and this sharing is greater than for paralogs.

Closest gene between 2 species are potential orthologous gene

Get orthologous assemblies

Reciprocal blast in general case

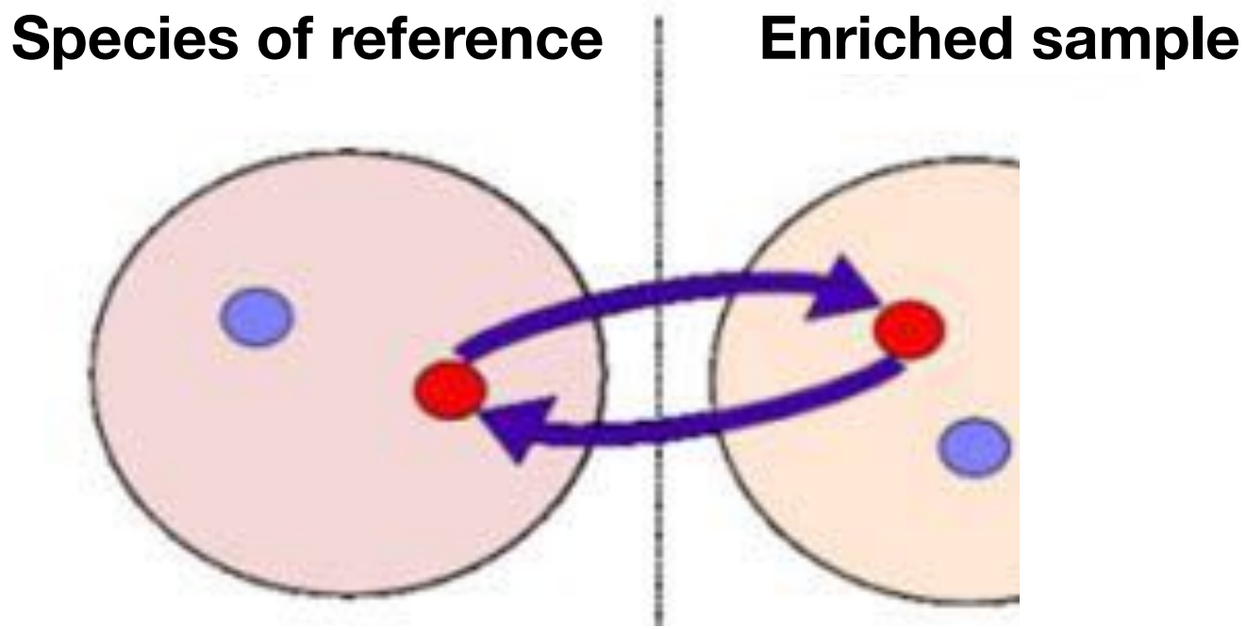
(1) sequence of gene which we want to find its orthologous sequence in other organisms
(2) genome of these 2 organisms



If a pair of genes in different species are the closest to each other, these 2 genes have "reciprocal best hit"

Get orthologous assemblies

Reciprocal blast in our pipeline

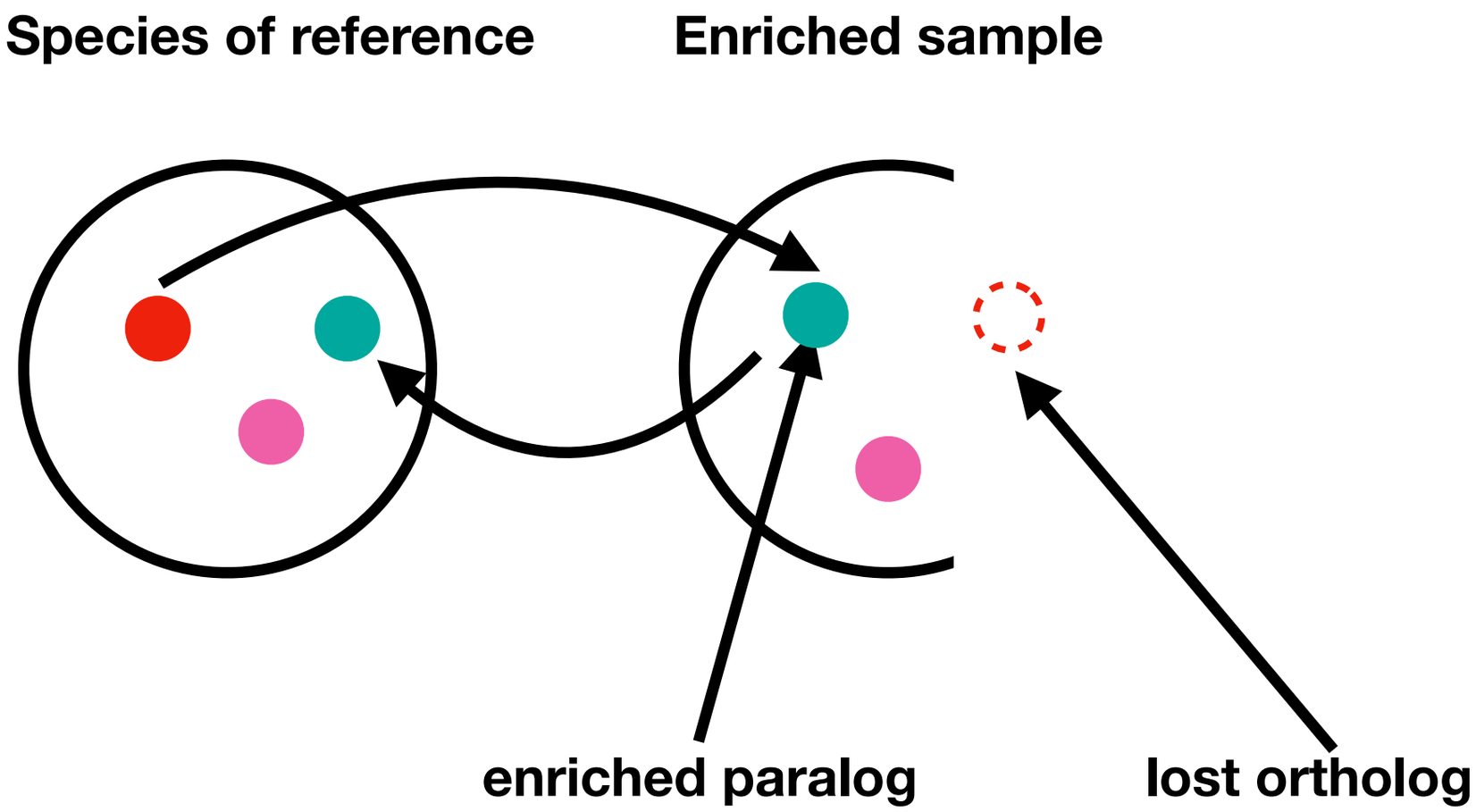


In our situation, we do not have the genome, only got several contigs only

But, loci of reference and contig have reciprocal best hit, then they are also putative orthologues

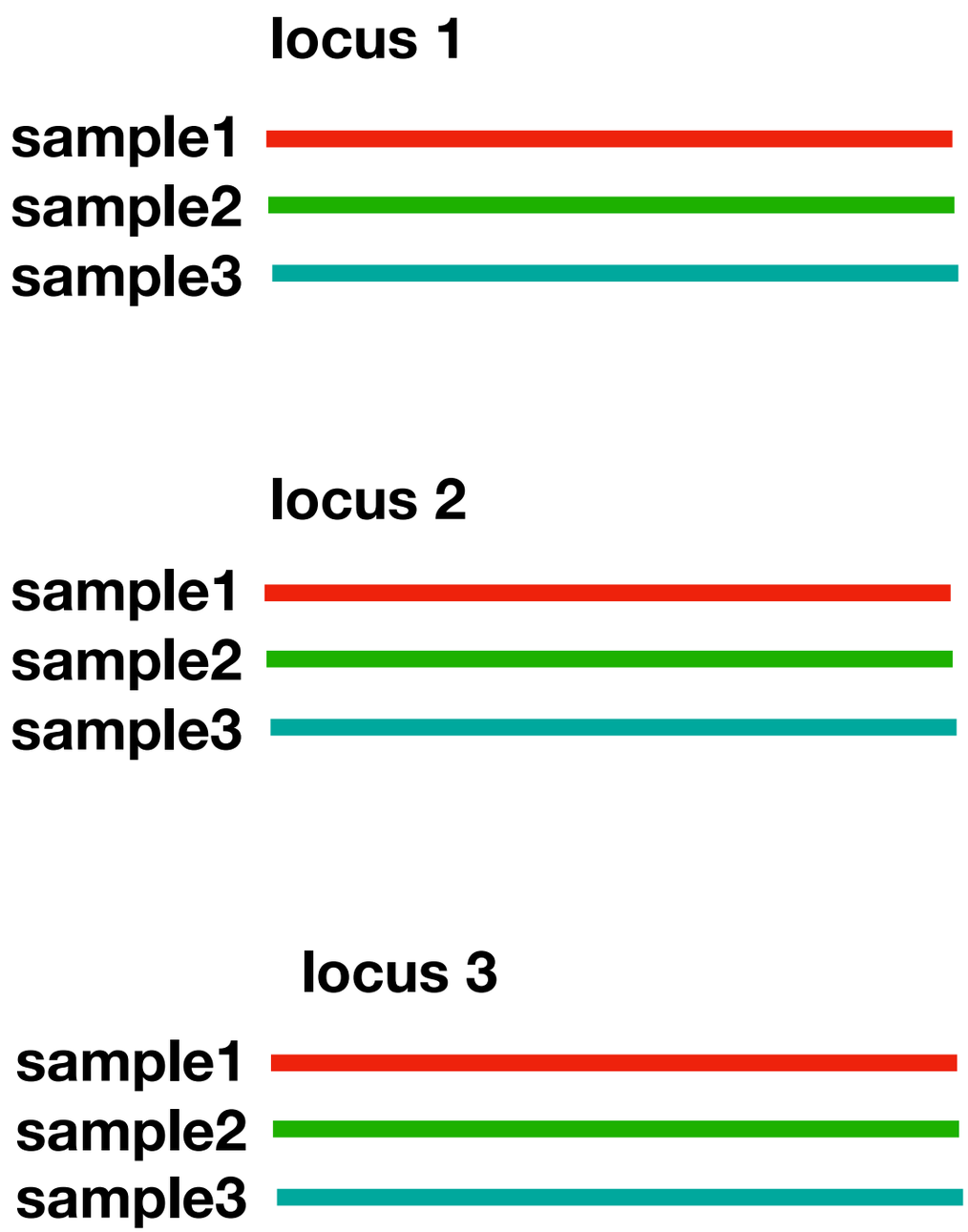
Get orthologous assemblies

Reciprocal blast in our pipeline



Exclude the contig if it does not have reciprocal blast hit with reference loci

Until here we've been reached our first goal



Output until here looks like this

A “>” before the sequence name

After name is sequence

```
>Oryzias_latipes.ultracontig115.4231577.4236646
TTAATATTGTGGACTTTTTGGAGGGATGCCGCATCCAGTCAAGTTCGTCAGGAGTTTCAGG
>CL130
TTAATATTGTGGACTTTTTGGAGGGATGCCGCAT
```

First one is the sequence of reference

Following is the sequence of enriched sample

This called fasta format. File suffix is “fa”, “fas” or “fasta”

We got to notice that enriched sequences are full-coding

Start from the first codon

End at from the third codon

TATAACCTG

Length of nucleotide can be exactly divided by 3

Y N L

No stop codon in amino acid sequences

Further processing

Align sequences

Filter bad aligned sequences

Summary statistics

Align sequences

Arranging the sequences of DNA, RNA, or protein to identify regions of similarity by insert gap (“-”)

Align sequences

```
Seq1    A T C G G C A G A
Seq2    A T C G G A G A
```

Qualify the similarity

score matrix
match = 1
mismatch = -1
linear gap = -1

Alignment 1

```
Seq1    A T C G G C A G A
Seq2    A T C G G  - A G A
```

Alignment 2

```
Seq1    A T C G G C A G A  -
Seq2    A T C G  -  -  G A G A
```

Align sequences

```
Seq1    A T C G G C A G A
Seq2    A T C G G A G A
```

Alignment 1

```
Seq1    A T C G G C A G A
Seq2    A T C G G  - A G A
```

score = 7

Alignment 2

```
Seq1    A T C G G C A G A  -
Seq2    A T C G  -  -  G A G A
```

score = -6

Alignment in our cases

```
>04TYPA
---------------GAGCCCACTATGGAGGACATACGGCGGATGCAGGCGGAGTTCACGGACGAGCGGGACTGGAATAAGTTTCACCAGCCTC
>05MIVE
--------------------ACTATGGAGGACATACGGCGAATGCAGGCGGAGTTCACCAACGAGCGGGACTGGAACAAGTTCCACCAGCCTC
>06GODO
TTTACTTTTACCTCCGAGCCCACTTTGGAGGACATACGGCGAATGCAGGCTGAGTTTACCGACGAGCGGGACTGGAATAAGTTTCACCAGCCTC
>09BUKO
TTCACCTTCAGCCCGGAGCCCACTATGGAGGACATACGGCGAATGCAGGCTGAGTTCACCGACGAGCGGGACTGGAACAAGTTTCACCAGCCCC
>1139-2
TTCACCTTCAGCCCCGAGCCCACTATGGAGGACATCAGGCAAATGCAGGCGGAGTTCACTGAAGAGCGGGACTGGAACAAGTTTCACCAGCCTC
>1149-10
TTCACCTTCAGCCCCGAACCCACTATGGAAGACATCAGGCAAATGCAGGCGGAGTTCACCGATGAGCGGGACTGGAACAAGTTTCACCAGCCTC
>1149-11
TTCACCTTCAGCCCCGAACCCACTATGGAAGACATCAGGCAAATGCAGGCGGAGTTCACCGATGAGCGGGACTGGAACAAGTTTCACCAGCCTC
>1149-9
TTCACCTTCAGCCCCGAACCCACTATGGAAGACATCAGGCAAATGCAGGCGGAGTTCACCGATGAGCGGGACTGGAACAAGTTTCACCAGCCTC
>1166-1
TTCAGC---AGCCCCGAGCCCACTATGGAGGACATCAGGCAAATGCAGGCGGAGTTCACTGAAGAGCGGGACTGGAACAAGTTTCACCAGCCTC
>1203
TTTACCTTCAGCCCCGAGCCCACCATGGAGGACATACGGCGAATGCAGGCGGAGTTCACCGACGAGCGGGACTGGAATAAGTTTCACCAGCCCC
```

Because sequences are full-coding, so gaps are also inserted in 3.
Length of alignment can be exactly divided by 3

Filter poorly aligned sequences

The resulting assemblies may still got some problems:

(1) chimera
(2) unidentified paralogs or unrelated sequence

Filter poorly aligned sequences

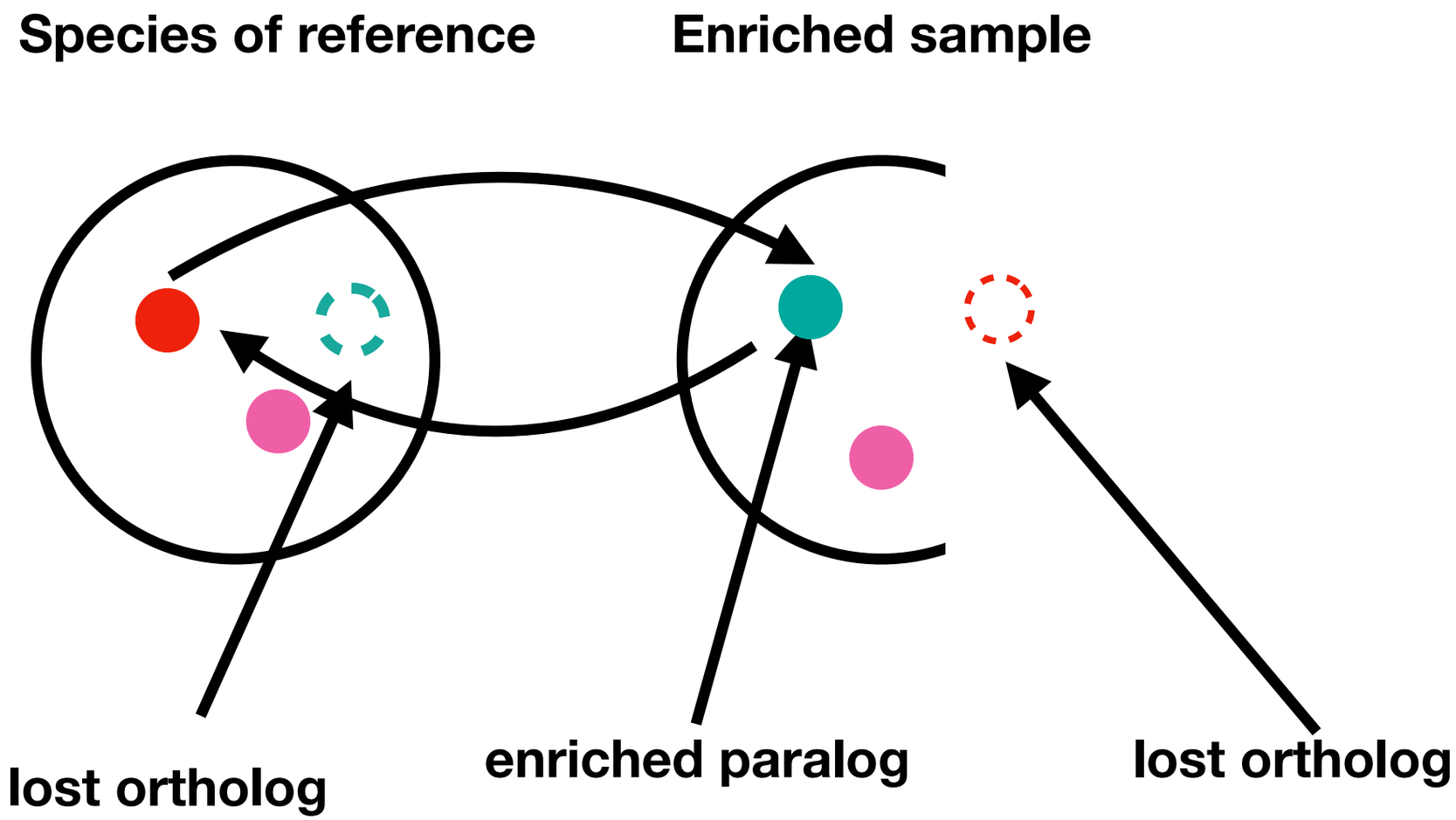
Chimera

```
-CATGCTCTCGACAATGACTCGGGTCCCTAT--------------------GGCCAGCTGACGTACTCCATTTTAACTTCCTGCTTCATGGAC-
-CAGGCTATTGACAATGACTCAGGCCCCTAT--------------------GGCCAGTTGACATACTCCATCTTAACATCCTGTTTTATGGAC-
CAAAACAACTGACAATGACTCAGGCCCCTAT--------------------GGCCAGCTGACATACTCCATCTTGACGTCCTGTTTTATGGAC-
-GCGTCAGTATTCACAGATATCGGGATGCTTCCAGATGAGGAGCTCCAAGGCGGGCGCGAGAGTTTGGCCACAGAGTCGAGCTGTCTCATGCACC
-CAAGCTATTGACAACGACGTAGGTCCGTAC--------------------GGTCAGCTAACGTACTCCATCTTGACGTCTTGCTTCATGGAC-
-CAAGCTATTGACAATGACGTAGGCCCTTAT--------------------GATCAGCTCACATACTCCATCTTGACCTCTTGCTTCATGGAC-
-AATGCTATTGACAATGACTCCGGCGTCTAT--------------------GGTCAGCTAACGTATTCCATCCTGACCTCCTGCTTCATGGAA-
-CACGCCATCGACAATGACTCTGGCCTCTAT--------------------GGGCAGCTAACATACTCCATCTTGACCTCCTGCTTCATGGAC-
-CATGCTCTAGACAATGACTTGGGTCCCTAT--------------------GGCCAGCTGACATACTCCATTTTAACTTCCTGCTTCATGGAC-
```

No sequence can be aligned with chimeric sequences

Filter poorly aligned sequences

unidentified paralogs or unrelated sequence



Filter poorly aligned sequences

How to filter unidentified paralogs

These paralogs are always too diverged from other captured sequences and reference

We use pairwise distance to measure the divergence

Filter poorly aligned sequences

What's pairwise distance

```
Seq1   A T C G G C A G A
Seq2   A T C C G  - A G A
```

Column with 2 bases = 8

Different base = 1

pairwise distance = 0.125

Other 2 kinds of special filters are also provided

Pick out loci follow the molecular clock hypotheses

Pick out loci have the provided monophyletic group

Detect contamination between taxa

Most of loci in diverged taxa cannot be very close

Detect contamination between taxa

Closely related group1: Human Chimp Orangutan

too close p-distance between taxa (<= 0.002)

Closely related group2: Tilapia Zebra fish

Contamination rate (%) = potentially contaminated pair/all pair*100

Summary statistics

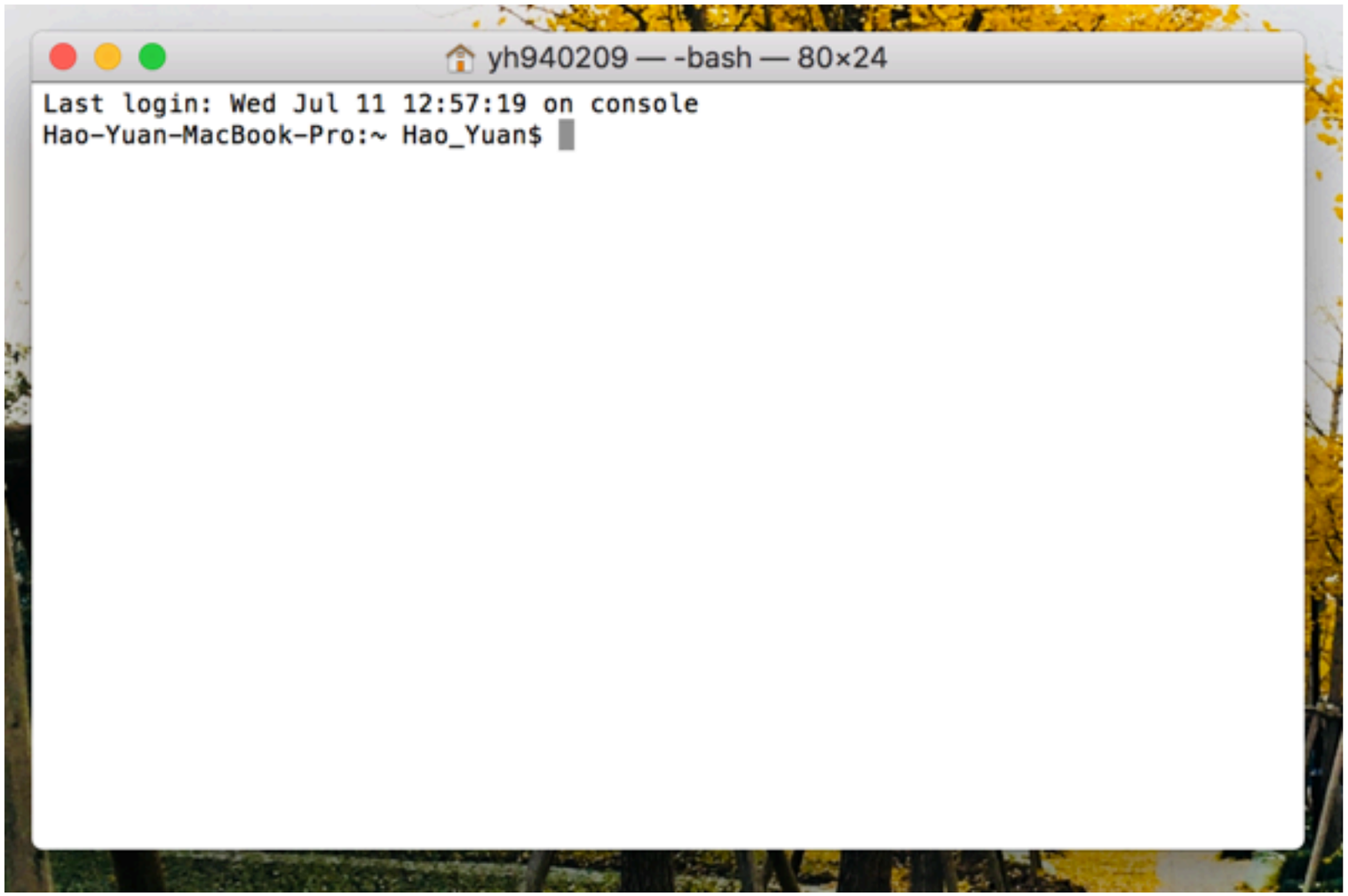
Summarized statistics for each locus including:
(1) Average length of coding region
(2) Average length of flanking region
(3) Length of alignment
(4) Average GC content
(5) Percentage of Missing data
(6) Pairwise distance

Summarized statistics for each sample including:
(1) Average length of captured sequences
(2) Average GC content
(3) Number of captured loci

Some prerequisites

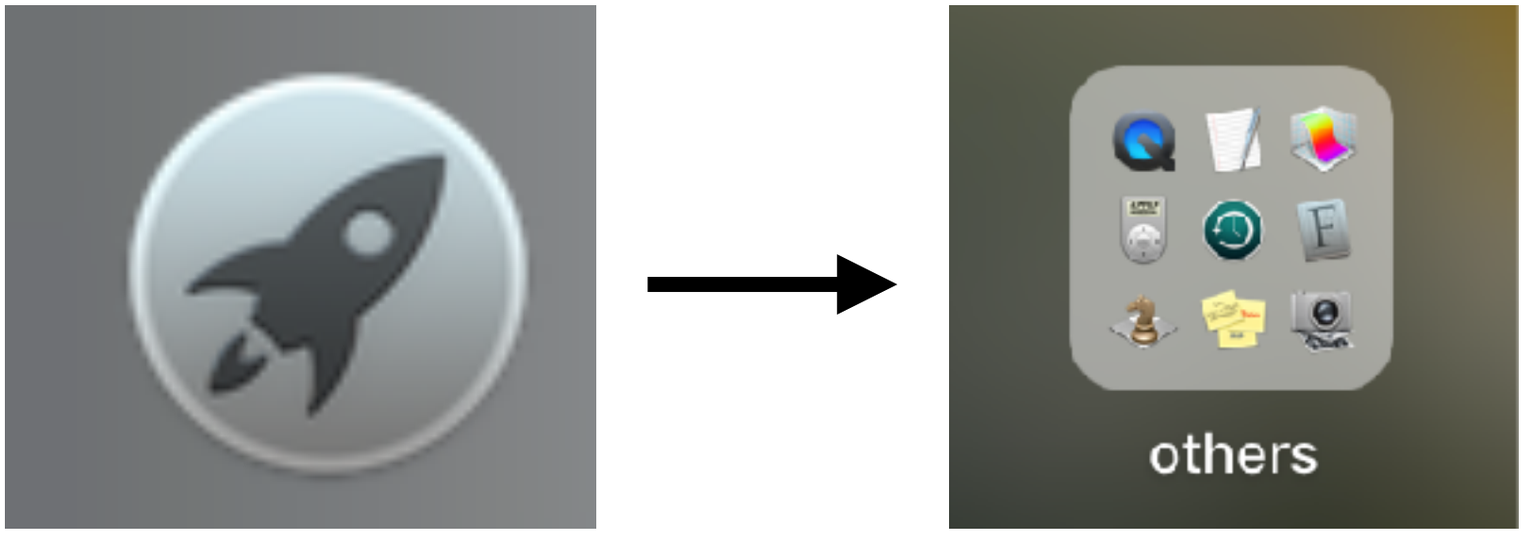
Tools we need

Terminal



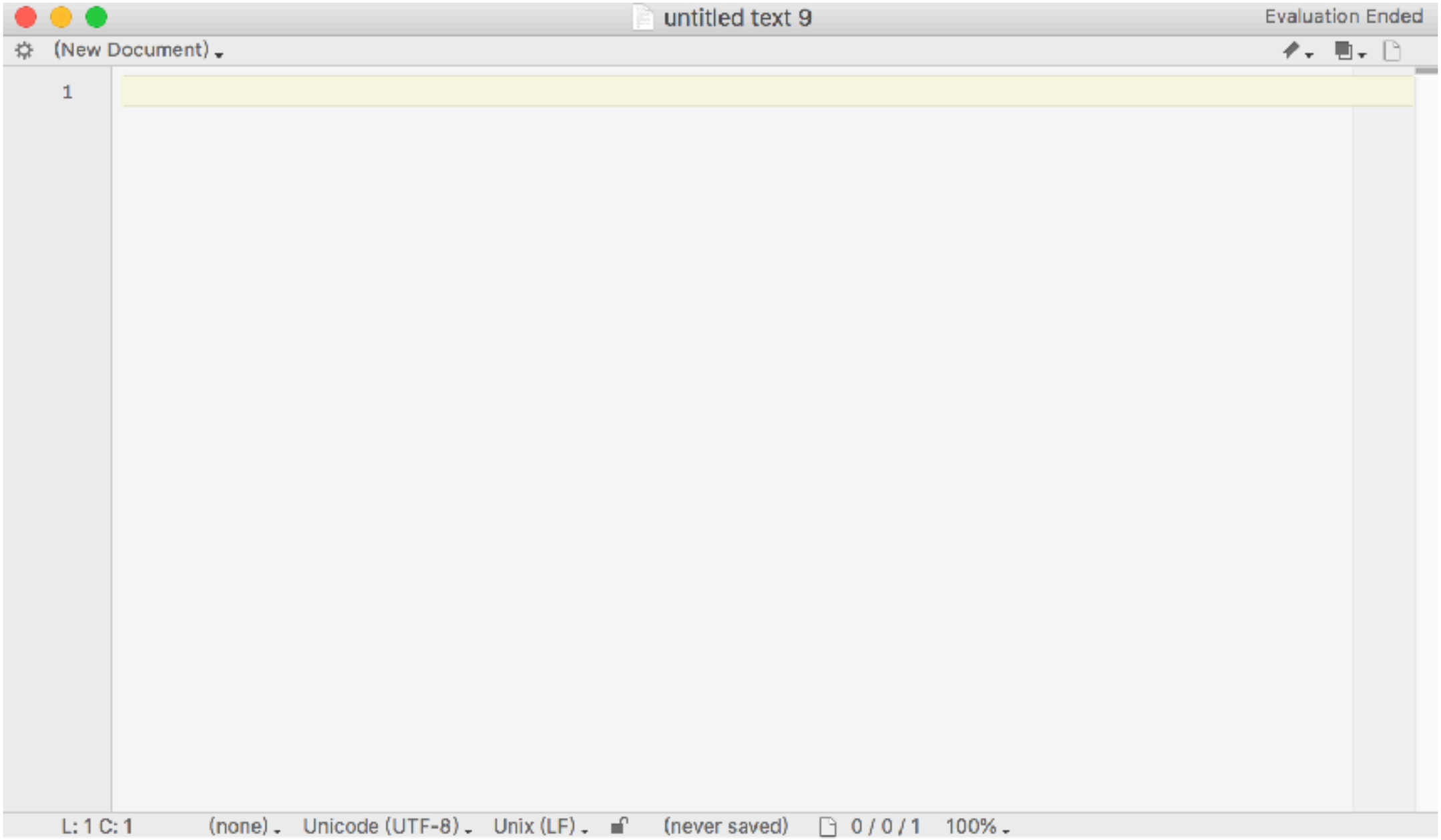
Tools we need

Terminal



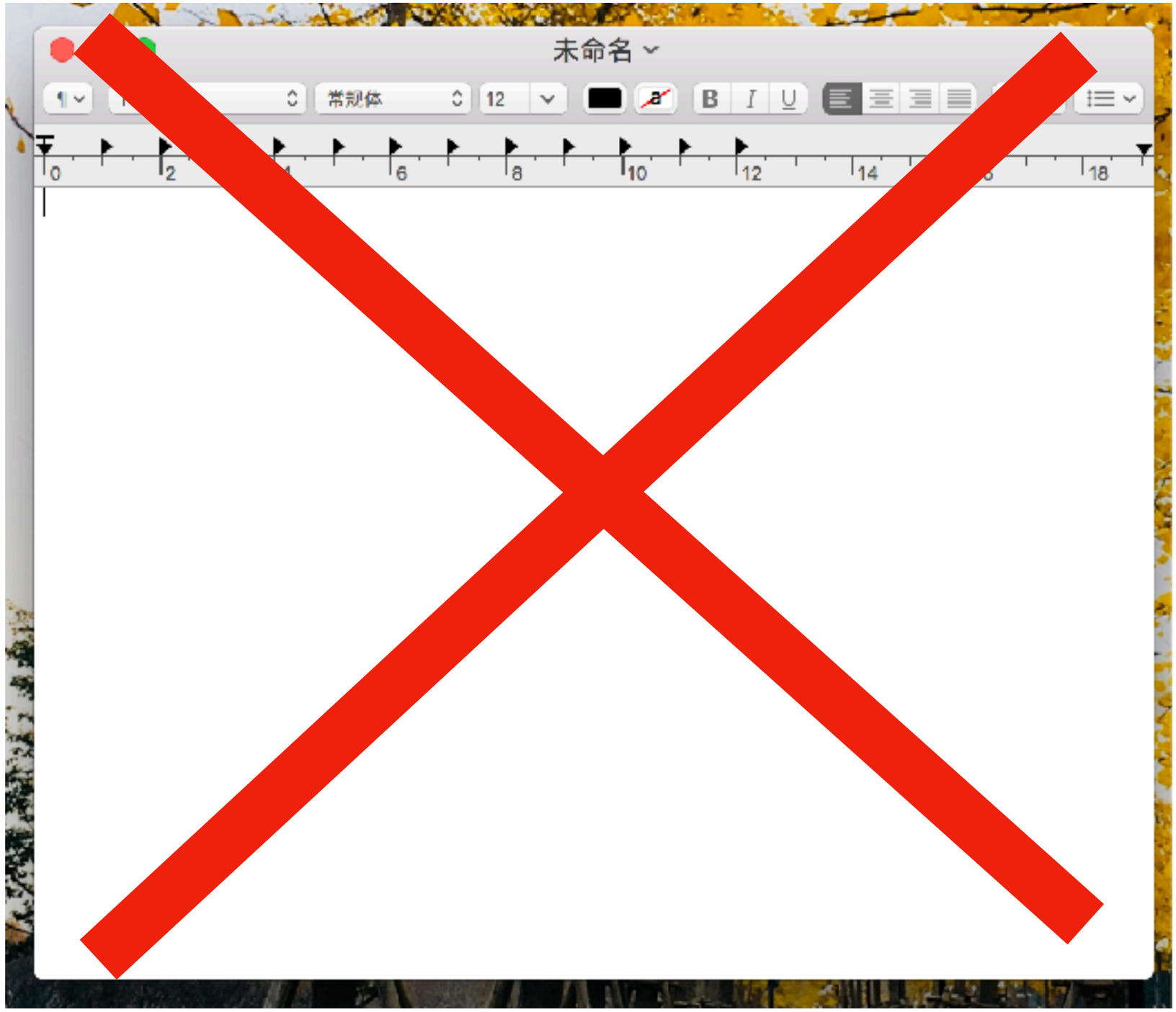
If you are using macOS, it can be found under “Launchpad->others”

Tools we need



A more professional but easy to use text editor

Tools we need



Simple text editor can only display plain text

Tools we need

What to do with text editor

Record commands during the analysis just like experiment records

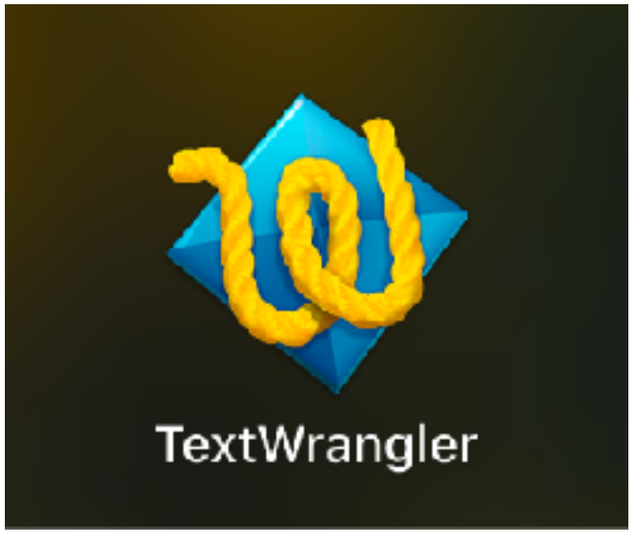
Write simple script

Check file format

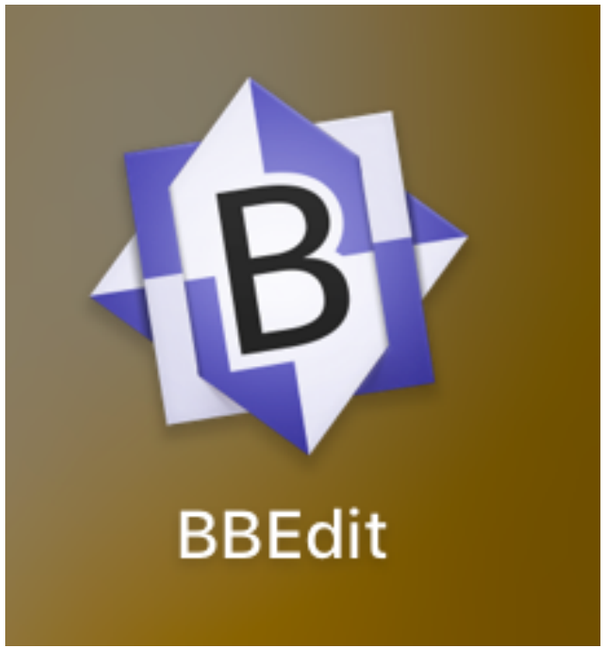
.....

Tools we need

A more professional text editor



TextWrangler (pre-installed on MacOS)



BBEdit (30-days trail)

Gedit (free)

Ultraedit (30-days trail)

Tools we need

A unix-like system

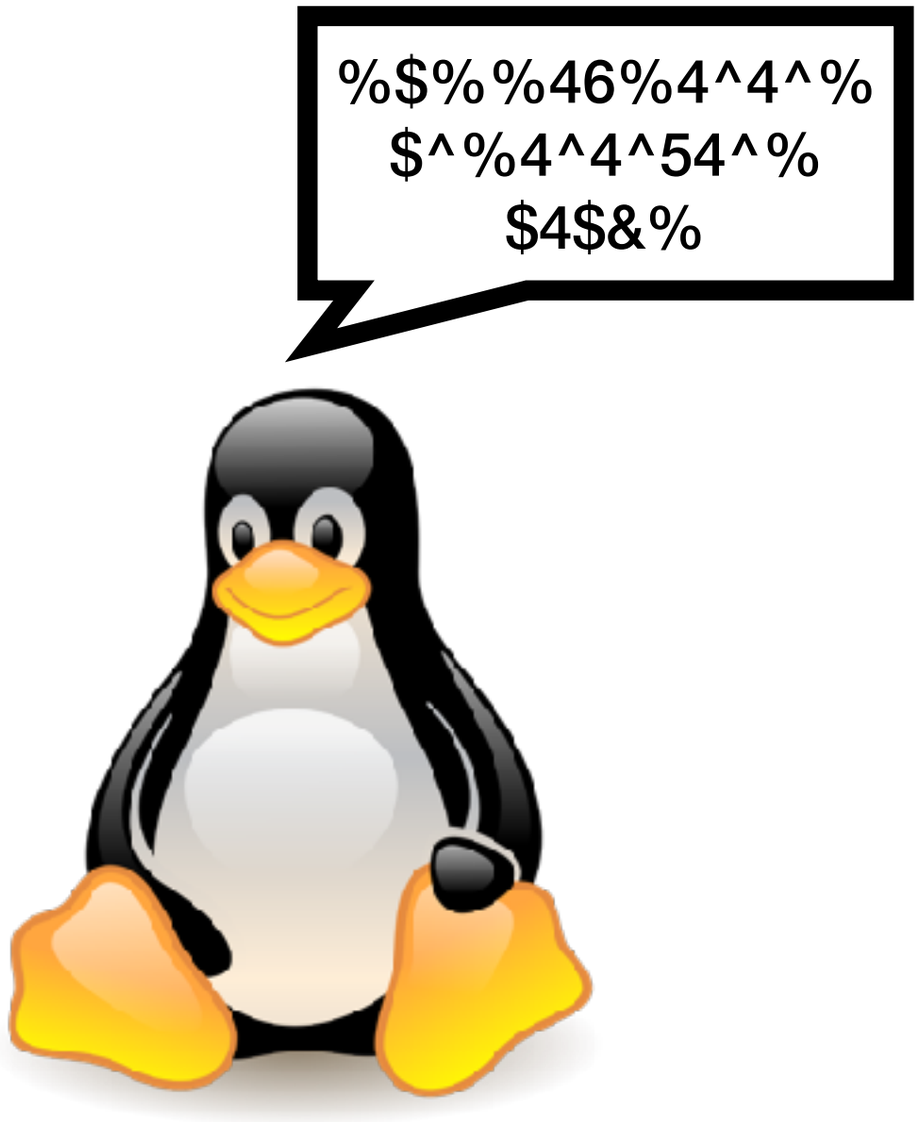
Linux

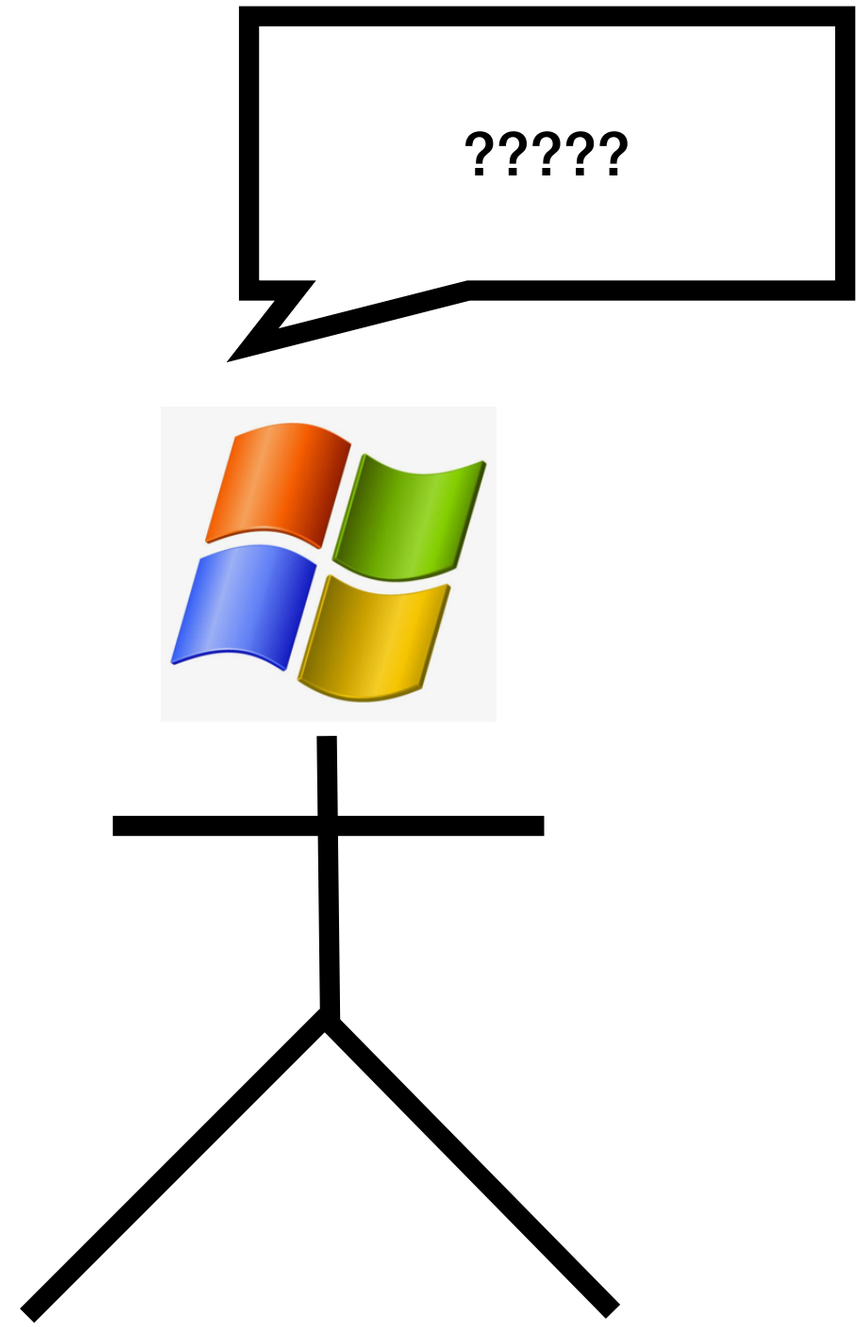
MacOS

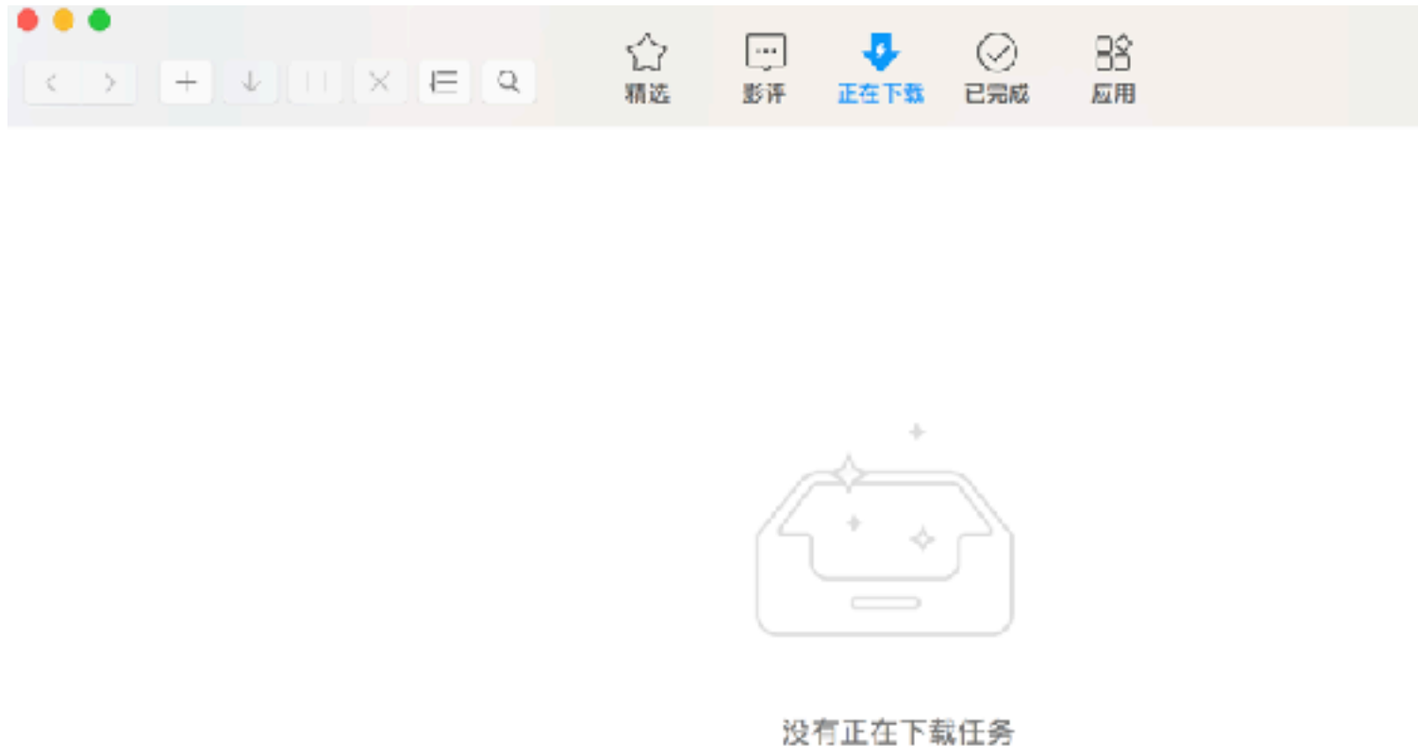
Tools we need

No windows here

Why no windows here







Graphical User Interface (GUI)

```
Last login: Mon Jul  9 14:57:36 on ttys000
Hao-Yuan-MacBook-Pro:~ Hao_Yuan$ cd Desktop/
```

```
Last login: Mon Jul  9 14:57:36 on ttys000
Hao-Yuan-MacBook-Pro:~ Hao_Yuan$ perl statistics.pl --nf_aligned test_normal_ali
gned
```

Command Line Interface (CLI)

How to type in and run a command in terminal

```
Last login: Mon Jul  9 15:27:04 on ttys000
Hao-Yuan-MacBook-Pro:Astral Hao_Yuan$
```

Name of your PC

The current directory

User name

Typed characters will be inserted here

command start after “$”

How to type in and run a command in terminal

```
Last login: Mon Jul  9 15:27:04 on ttys000
Hao-Yuan-MacBook-Pro:Astral Hao_Yuan$
```

Typed characters will be inserted here

```
Last login: Mon Jul  9 16:12:43 on ttys003
Hao-Yuan-MacBook-Pro:Astral Hao_Yuan$ ls
```

How to type in and run a command in terminal

```
Last login: Mon Jul  9 16:12:43 on ttys003
Hao-Yuan-MacBook-Pro:Astral Hao_Yuan$ ls
```

Type “enter”

Some output from the command

```
Hao-Yuan-MacBook-Pro:Astral Hao_Yuan$ ls
README.md                lib                          thesis-astral.pdf
astral.4.11.1.jar        test_data
Hao-Yuan-MacBook-Pro:Astral Hao_Yuan$
```

A new line to type in other command

In the following slides, i will use “$” to indicate the start of a command, like:

```
$ perl filter.pl --indir re_od_nogap --filtered filtered --cpu 4
```

DO NOT INCLUDE “$” INTO YOUR COMMAND

General format of a command

Format 1:

$ /path/to/software/name_of_software **Just the path to software**

Format 2:

$ /path/to/software/name_of_software option **Path to software + option**

option are used to pass parameter to software

Some option are mandatory like name of input file, while the others are optional like some parameter

General format of a command

Some variation of format 1 :

$ /path/to/software/name_of_software	call the software in complete path
$ ./name_of_software	call the software in relative path, the software is at current directory
$ ../name_of_software	call the software in relative path, the software is at parent directory

Some variation of format 2 :

$ ./name_of_software -option1 value_of_option	pass the value to software through option “option1”
$ ./name_of_software value_of_option	pass the value directly to the software

Environment variable

How to use a software without specifying the path to it ?

Environment variable

How to use a software without specifying the path to it ?

They are series of pre-set value. They will be called when you open the terminal. One of the example is $PATH

$PATH includes lots of user-defined path

Environment variable

```
export PATH=$PATH:/Users/yh940209/Programs/sqlite/
export PATH=$PATH:/Users/yh940209/Programs/bpp3.2/
export PATH=$PATH:/Users/yh940209/Programs/
export PATH=$PATH:/Users/yh940209/Programs/trinityrnaseq-2.2.0
export PATH=$PATH:/Users/yh940209/Programs/BCFtools/bin
```

Path to directory containing a software called bcftools

Environment variable

```
export PATH=$PATH:/Users/yh940209/Programs/sqlite/
export PATH=$PATH:/Users/yh940209/Programs/bpp3.2/
export PATH=$PATH:/Users/yh940209/Programs/
export PATH=$PATH:/Users/yh940209/Programs/trinityrnaseq-2.2.0
export PATH=$PATH:/Users/yh940209/Programs/BCFtools/bin
```

Path to directory containing a software called bcftools

$ /Users/yh940209/Programs/BCFtools/bin/bcftools -h

$ bcftools -h

When path is not specified before the software, system will find software under path saved in $PATH

Environment variable

$ cd	change directory to home (~)
$ nano .bash_profile	open file “.bash_profile” by an text editor called “nano”, it will create a new file if it does not exist
	type “export…” as showed in previous slide
	ctrl+o to write out the file
$ source .bash_profile	Reload $PATH, or you can simply reopen the terminal
$ echo $PATH	print the directory saved in $PATH on the terminal

Authority

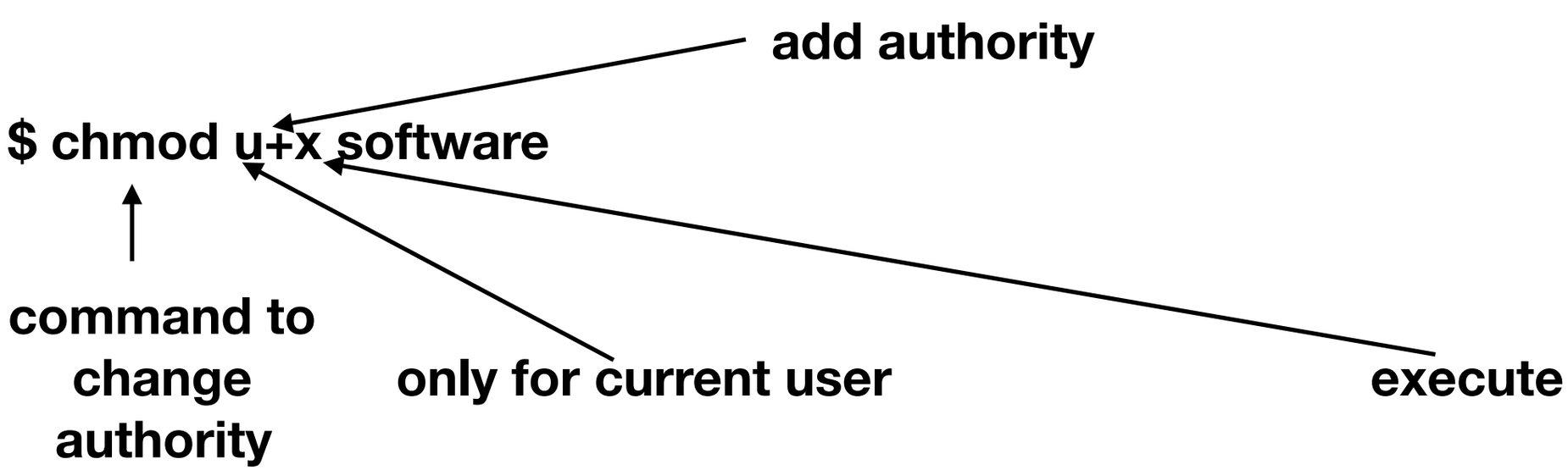
Each user have 3 kinds of authority to a file :

read (r)

write (w)

execute (x)

If you want to make a script or software executable for current user, type:



How to run a perl code

$ perl assemble.pl	Use the perl interpreter found under $PATH to run the code “assemble.pl”
$ perl assemble.pl -h	pass option “h” (help message) to the code
$ chmod u+x assemble.pl $ ./assemble.pl	You can call this way only if assemble.pl is executable
$ assemble.pl	Call assemble.pl found under $PATH. In this situation, assemble.pl must be executable

Dependencies

Software

Module

How to install a software

ncbi-blast-2.7.1+-src.tar.gz

source code

ncbi-blast-2.7.1+-x64-macosx.tar.gz

ncbi-blast-2.7.1+-x64-linux.tar.gz

ncbi-blast-2.7.1+-1.x86_64.rpm

binary

ncbi-blast-2.7.1+.dmg

Installer

How to install a software

ncbi-blast-2.7.1+-src.tar.gz

Compiler

source code → **binary**

ncbi-blast-2.7.1+-x64-macosx.tar.gz

ncbi-blast-2.7.1+-x64-linux.tar.gz

ncbi-blast-2.7.1+-1.x86_64.rpm

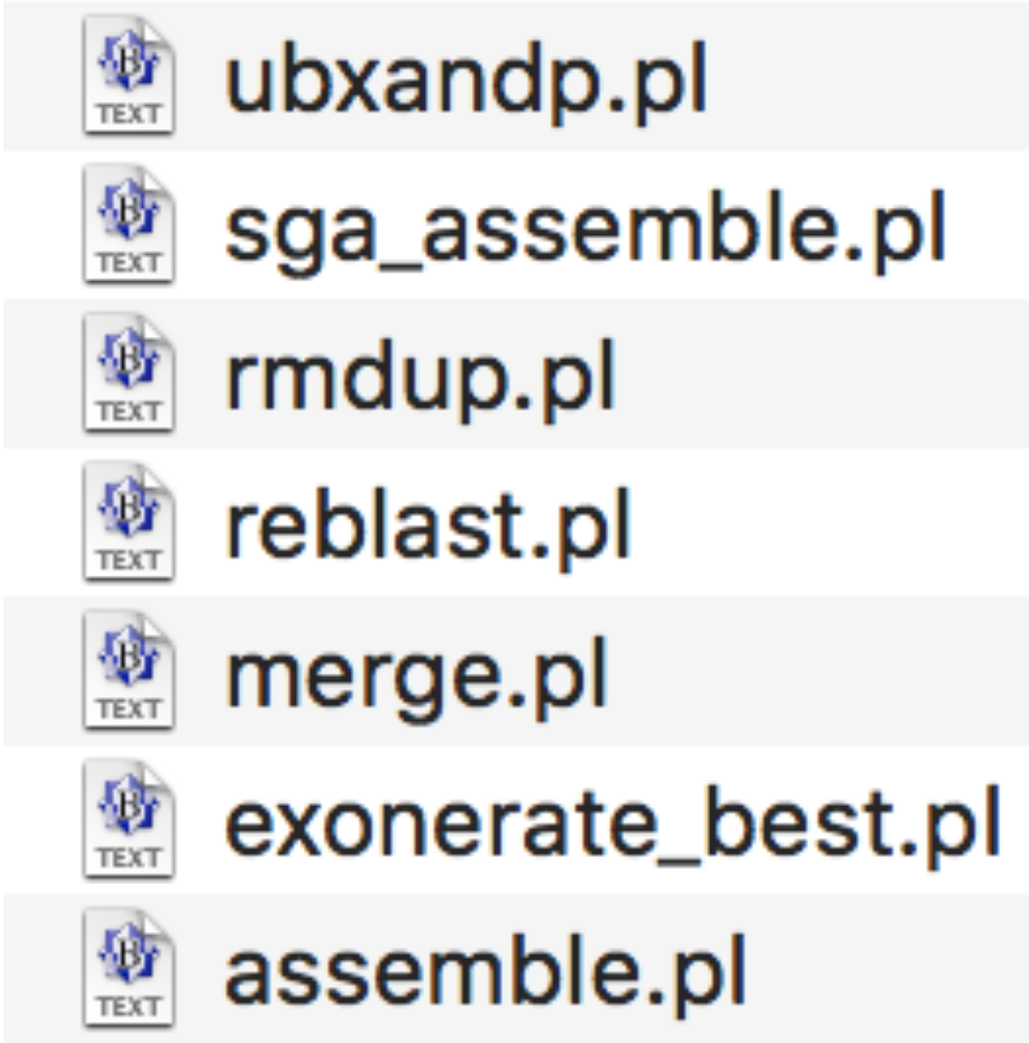
ncbi-blast-2.7.1+.dmg

binary

Installer → **binary**

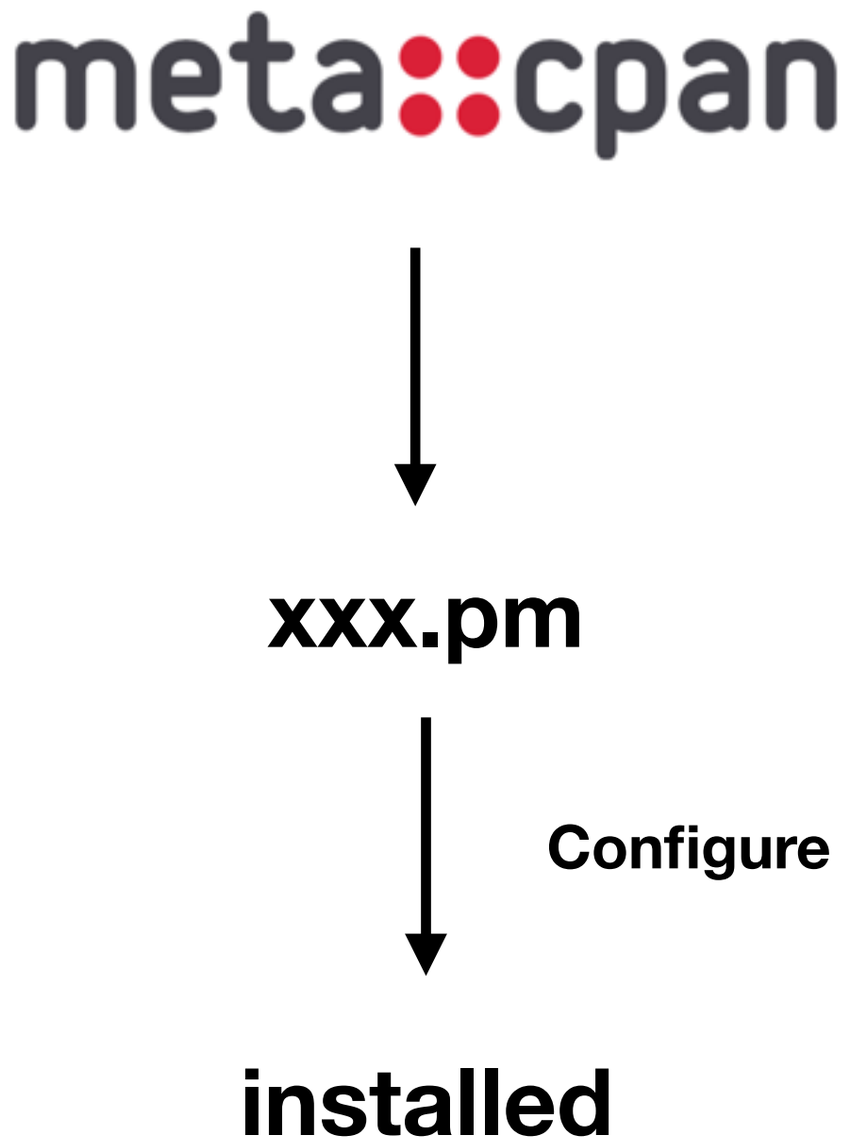
Different system varies

How to install a software



Download and use source code directly

How to install module



How to install module

xxx1.pm

xxx2.pm

xxx.pm ←

xxx3.pm

xxx4.pm

if a modules needs lots of other modules ?

How to install module

```
$ perl -MCPAN -e shell
```

or

```
$ sudo perl -MCPAN -e shell
```

sudo allows a permitted user to
execute a command as the superuser

How to install module

```
Hao-Yuan-MacBook-Pro:~ Hao_Yuan$ perl -MCPAN -e shell
```

How to install module

```
Hao-Yuan-MacBook-Pro:~ Hao_Yuan$ perl -MCPAN -e shell
```

```
Hao-Yuan-MacBook-Pro:~ Hao_Yuan$ perl -MCPAN -e shell
Terminal does not support AddHistory.

cpan shell -- CPAN exploration and modules installation (v2.11)
Enter 'h' for help.

cpan[1]>
```

How to install module

```
cpan[1]> install R.pm
Reading '/Users/yh940209/.cpan/Metadata'
  Database was generated on Mon, 09 Jan 2017 04:53:40 GMT
Reading '/Users/yh940209/.cpan/sources/authors/01mailrc.txt.gz'
.....................................................................DONE
Fetching with HTTP::Tiny:
http://cpan.communilink.net/modules/02packages.details.txt.gz
```

type “install xxx.pm” and “enter”

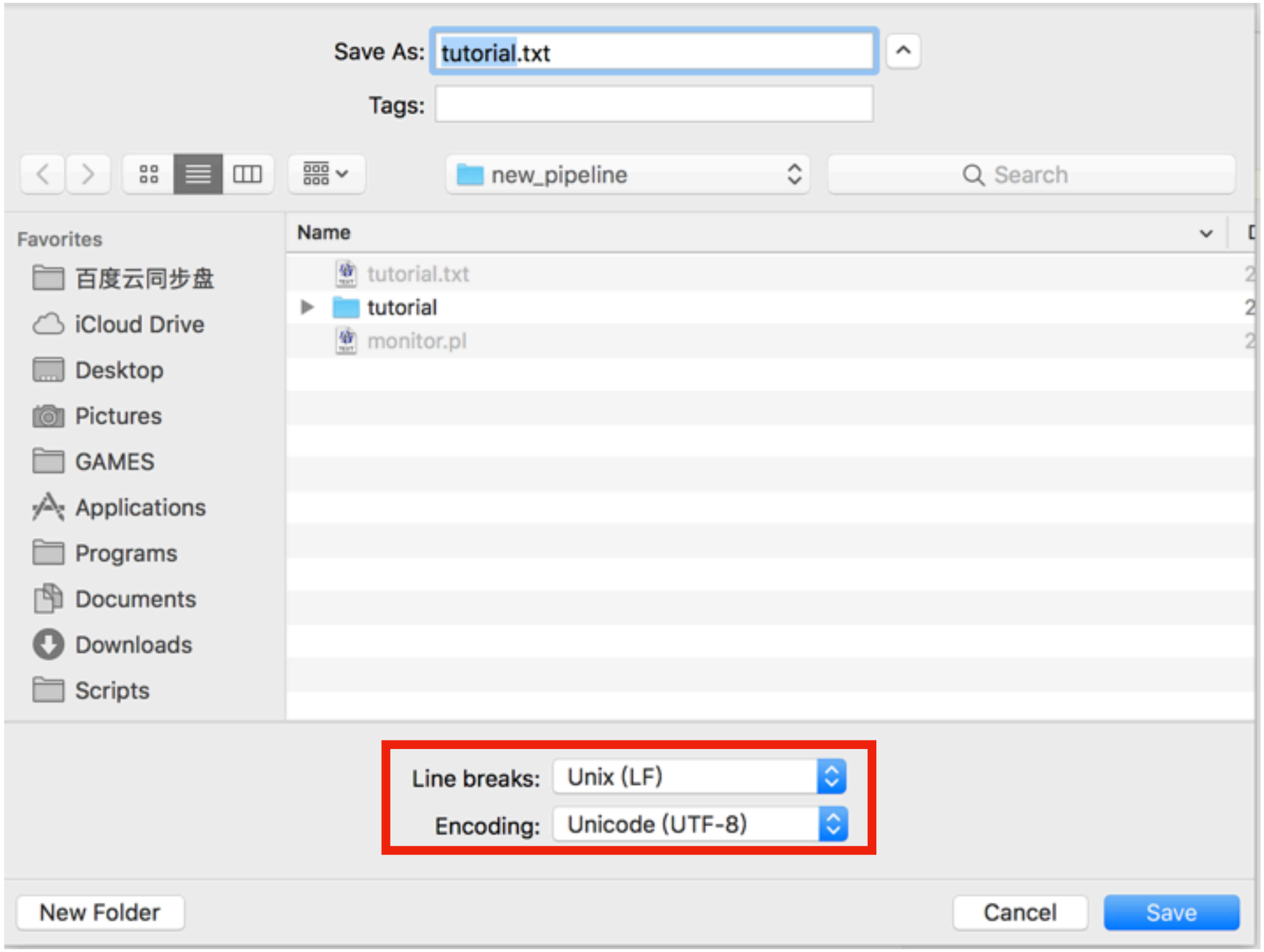
Wait a few minute.The current installing module and its related module will be installed together

Format of infile

Every files under analysis must be encoded by unicode (utf-8) and line breaks is unix (LF).

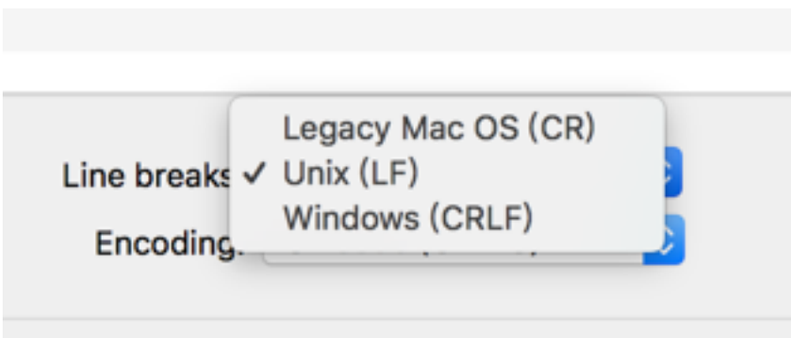
Format of infile

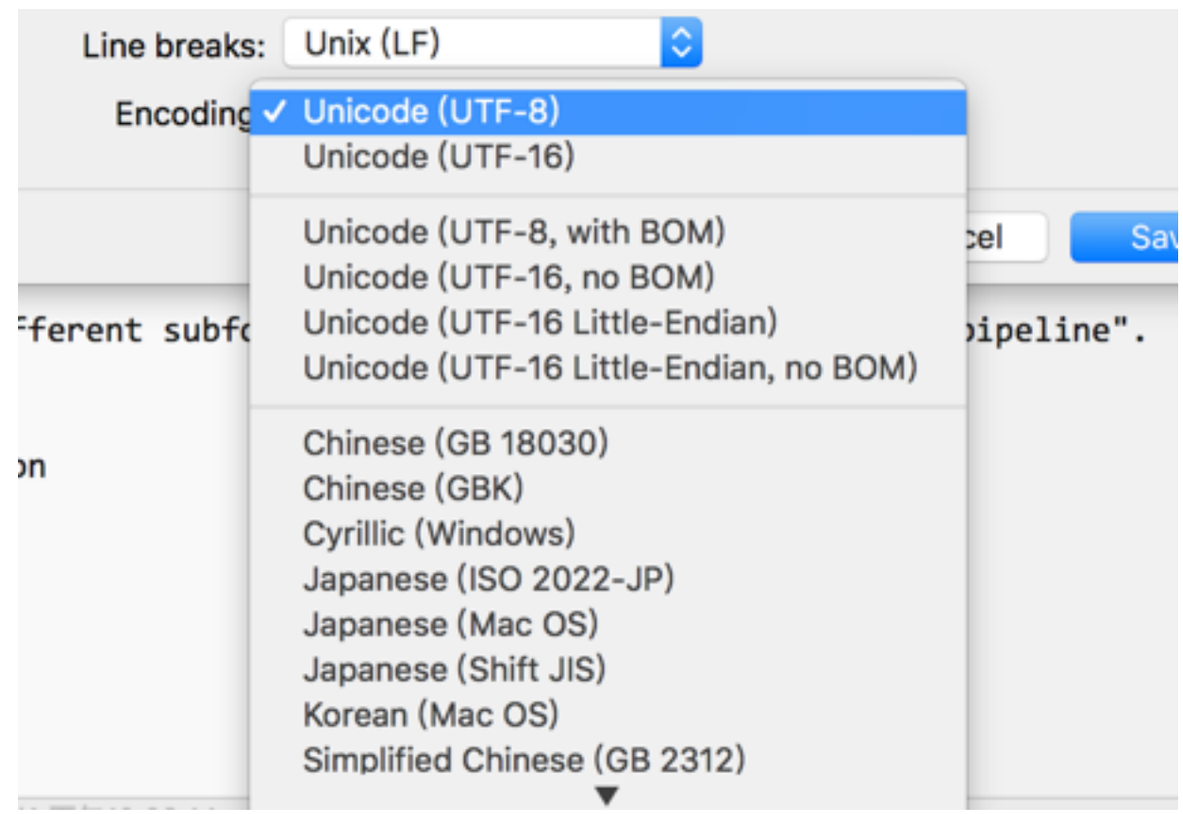
How to check encoding and line break



Format of infile

Encoding and line break





Check encoding and line breaks especially files were produced or saved on windows

Format of infile

Avoid using non-English charaters, space or any other strange characters like "#?@!" in filename

If you want use space, substitute it as “_” (underline)

We will start to learn how to assemble sequences get through target enrichment

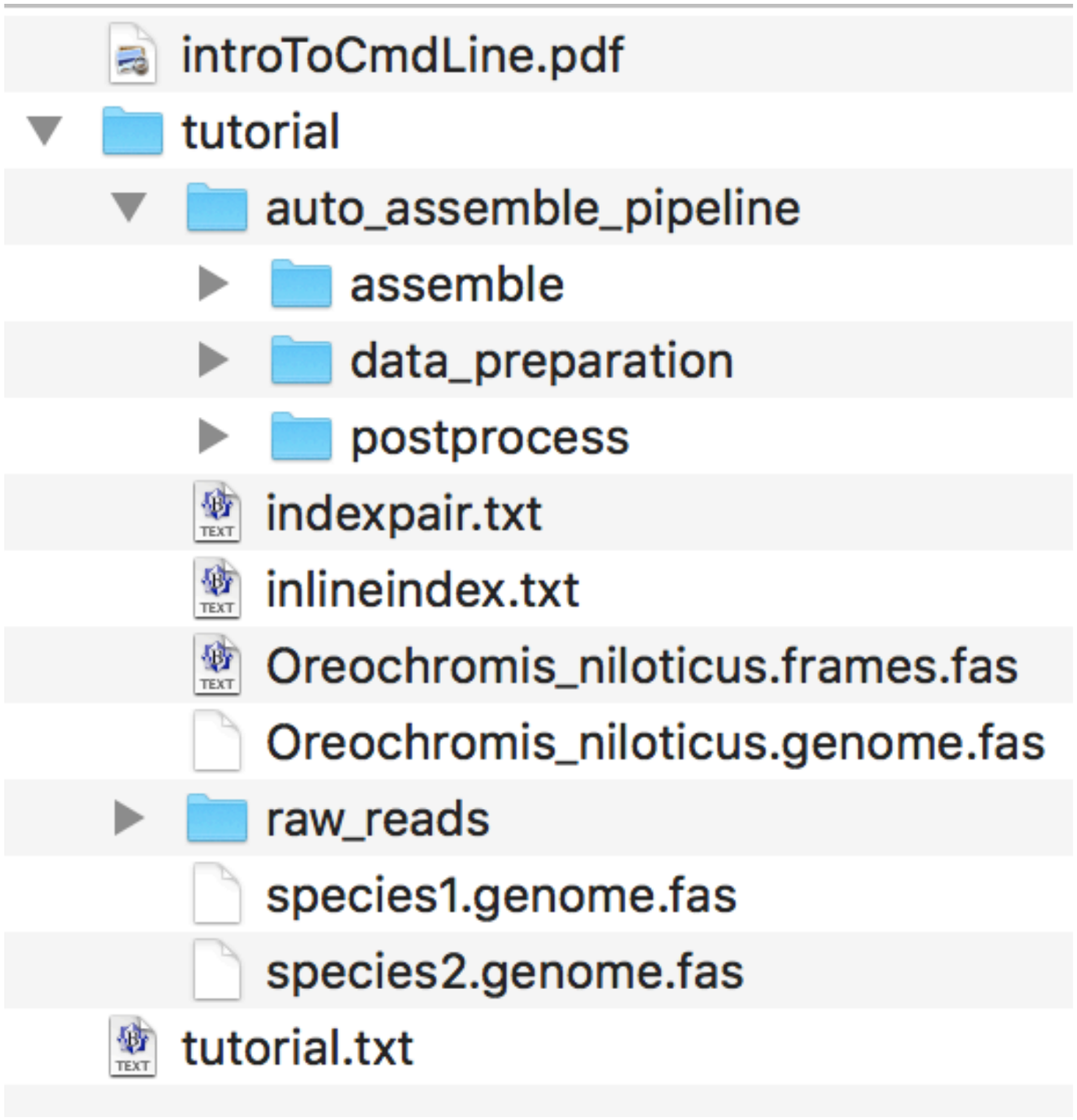
In following tutorial, each script will be introduced in this way:

(1) Function
(2) dependencies
(3) Usage
(4) involved option
(5) Input and output

Due to time limit, i won't introduce each option and script in detail.
All scripts included in the pipeline and its function are briefly introduced at line 70-117 of "tutorial.txt"

Access its detailed usage by:
$ perl name_of_script.pl -h

Learn more about command line, please refer to "introToCmdLine.pdf" under gzipped package "toturial_test_data.tar.gz"



Introduction to command line

All scripts

Test data

Tutorial

We've already been in running directory (/home/users/cli/ocean/yuanhao/pipeline_demonstration/toturial_test_data/tutorial). Let's see what's under it by:

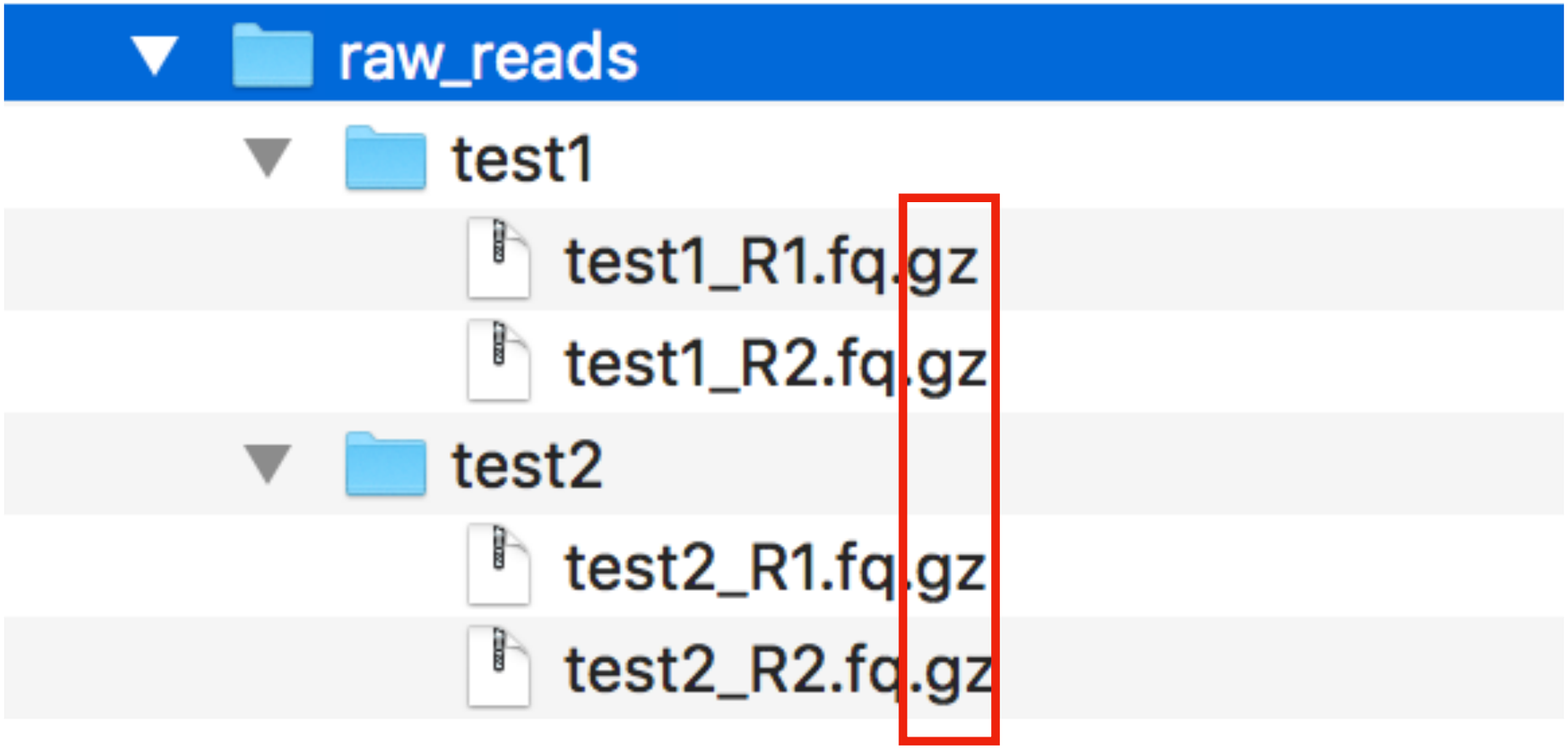
$ ls

command to list what's under a directory

```
-bash-4.1$ ls
Oreochromis_niloticus.frames.fas  indexpair.txt    species1.genome.fas
Oreochromis_niloticus.genome.fas  inlineindex.txt  species2.genome.fas
auto_assemble_pipeline            raw_reads
```

Let’s start run the scripts!

Data preparation



Reads are compressed

Gunzip data

Dependencies: no

Input: raw_reads

Usage:

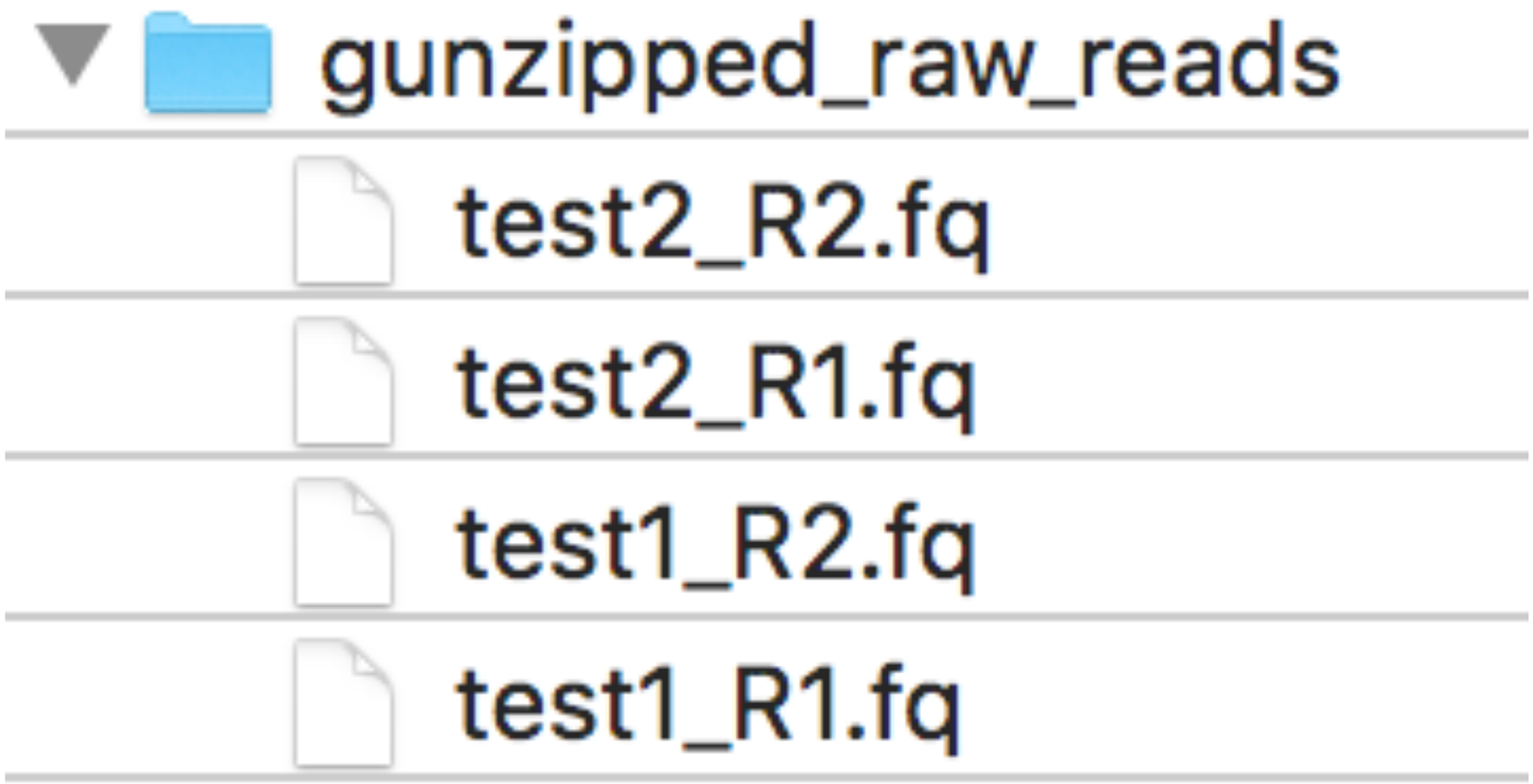
"\" is used to continue the command line in a new line

```
$ perl ./auto_assemble_pipeline/data_preparation/gunzip_Files.pl \
--gzip raw_reads \
--gunzipped gunzipped_raw_reads
```

Involved options:

--gzip: Directory containing gzipped raw data
--gunzipped: Directory containing expanded raw data

Output:
gunzipped_raw_reads: Directory containing expanded raw data


gunzipped_raw_reads
test2_R2.fq
test2_R1.fq
test1_R2.fq
test1_R1.fq

Demultiplex sample based on inline index (optional)

Dependencies: no

Input:
gunzipped_raw_reads
inlineindex.txt
indexpair.txt

```
$ perl ./auto_assemble_pipeline/data_preparation/demultiplex_inline.pl \
--undemultiplexed gunzipped_raw_reads \
--demultiplexed demultiplexed \
--inline_index inlineindex.txt \
--index_pair indexpair.txt
```

Involved options:

--undemultiplexed: Directory containing expanded raw data
--demultiplexed: Directory containing demultiplexed raw data
--inline_index: File records the sequences and number of inline index.
--index_pair: File records pairs of inline index for each sample.

Output:
demultiplexed: Directory containing demultiplexed raw data
unpaired_reads: Directory containing unpaired reads for each sample

inlineindex.txt

Name

IS1 adaptor + inline index

Index	Name	Sequence	Name	Sequence
TCTGCC	IS1_Ind1	A*C*A*C*TCTTTCCCTACACGACGCTCTTCCGATCTtc*t*g*c*c	IS3_Ind1	ggcagaAGATCGGAA*G*A*G*C
GTCTCT	IS1_Ind2	A*C*A*C*TCTTTCCCTACACGACGCTCTTCCGATCTgt*c*t*c*t	IS3_Ind2	agagacAGATCGGAA*G*A*G*C
ATATTG	IS1_Ind3	A*C*A*C*TCTTTCCCTACACGACGCTCTTCCGATCTat*a*t*t*g	IS3_Ind3	caatatAGATCGGAA*G*A*G*C
TGGAAG	IS1_Ind4	A*C*A*C*TCTTTCCCTACACGACGCTCTTCCGATCTtg*g*a*a*g	IS3_Ind4	cttccaAGATCGGAA*G*A*G*C
TCTAGT	IS1_Ind5	A*C*A*C*TCTTTCCCTACACGACGCTCTTCCGATCTtc*t*a*g*t	IS3_Ind5	actagaAGATCGGAA*G*A*G*C
AGAGTA	IS1_Ind6	A*C*A*C*TCTTTCCCTACACGACGCTCTTCCGATCTag*a*g*t*a	IS3_Ind6	tactctAGATCGGAA*G*A*G*C
GGCCAA	IS1_Ind7	A*C*A*C*TCTTTCCCTACACGACGCTCTTCCGATCTgg*c*c*a*a	IS3_Ind7	ttggccAGATCGGAA*G*A*G*C
TATCTC	IS1_Ind8	A*C*A*C*TCTTTCCCTACACGACGCTCTTCCGATCTta*t*c*t*c	IS3_Ind8	gagataAGATCGGAA*G*A*G*C

6 bp sequence of inline index

IS3 adaptor + inline index

indexpair.txt

Name of demultiplexed sample

Name of undemultiplexed sample

yang2-01	CL452_1	1	25
	CL452_2	2	26
	CL452_3	3	27
	CL452_4	4	28
	CL452_5	5	29
	CL452_6	6	30
	CL452_8	7	31
	CL452_9	8	32

Corresponding inline index id

Must write in the order :
IS1 index IS3 index

Save as txt and notice about the encoding format (Unicode utf-8) and line break (Unix (LF))

Trim adaptor and low quality bases

When inline index are involved in samples

Dependencies:
(1) trim_galore
(2) cutadapt

Input:
(1) demultiplexed
(2) inlineindex.txt
(3) indexpair.txt

Usage:

```
$ perl ./auto_assemble_pipeline/data_preparation/trim_adaptor.pl \
--demultiplexed demultiplexed \
--inline_index inlineindex.txt \
--index_pair indexpair.txt \
--trimmed trimmed
```

Involved options:
--demultiplexed: Directory containing demultiplexed raw data
--inline_index: File records the sequences and number of inline index.
--index_pair: File records pairs of inline index for each sample.
--trimmed: Output directory containing adaptor and low quality bases trimmed reads

Output:
trimmed: Directory containing reads without adaptor and low quality bases
trimming_report: Directory containing trimming report for each sample
trimmed_reads_bases_count.txt: Tab delimited table recording number of reads and bases of raw data and trimmed data

When no inline index are involved in samples

Input: gunzipped_raw_reads

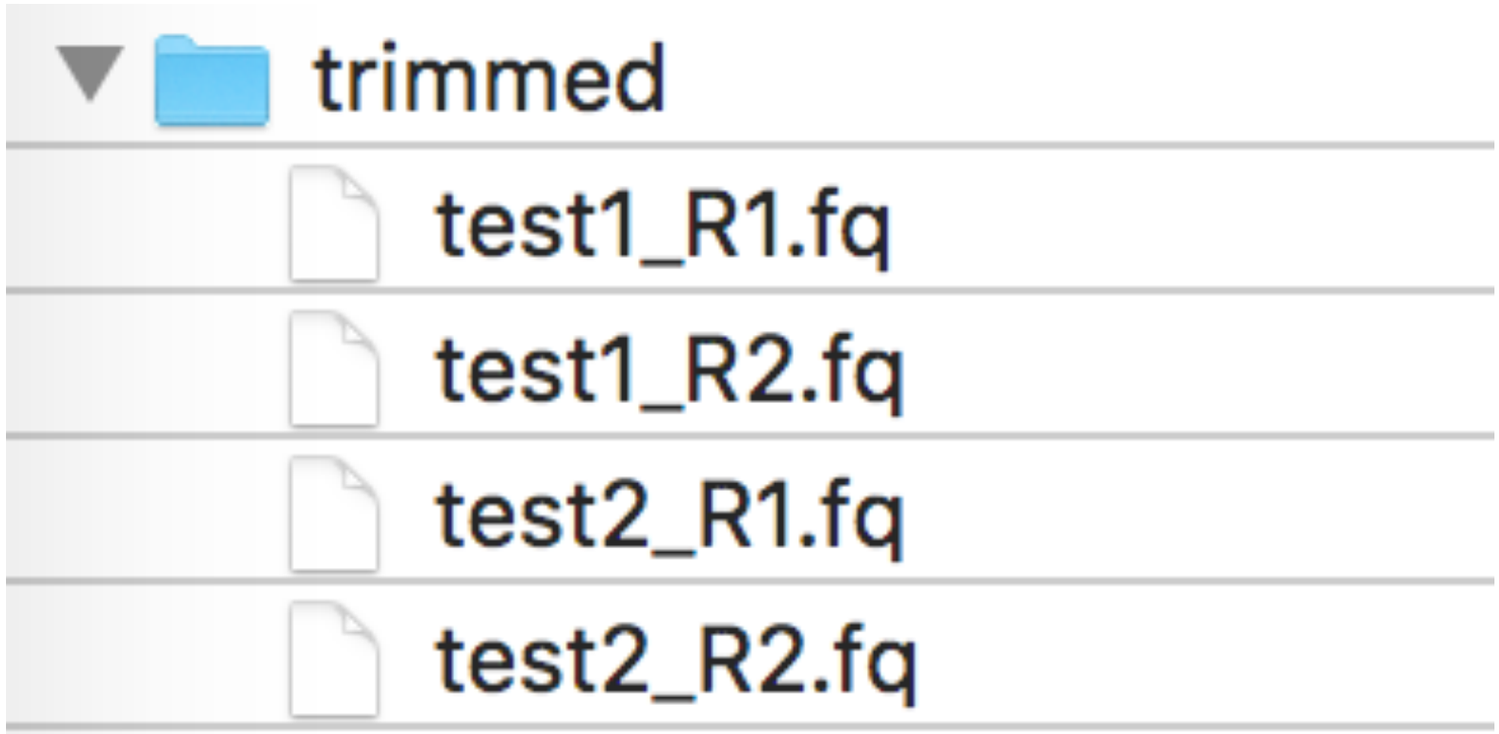
Usage:

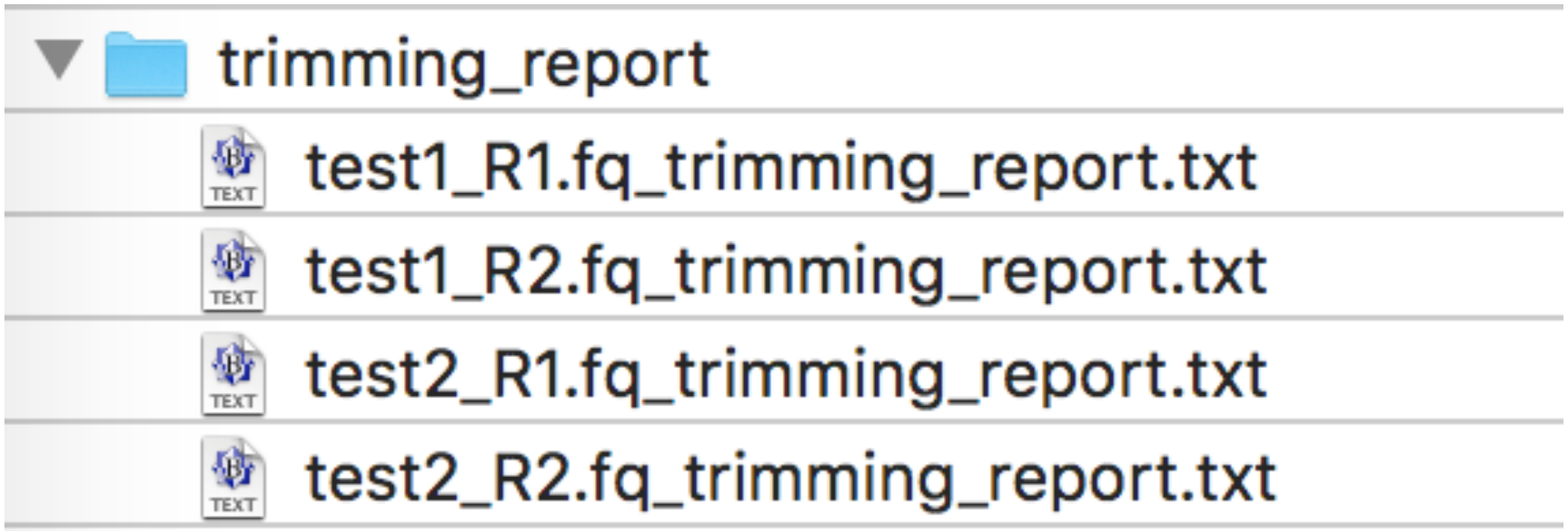
```
$ perl ./auto_assemble_pipeline/data_preparation/trim_adaptor.pl \
--demultiplexed gunzipped_raw_reads \
--trimmed trimmed
```

Output:
trimmed: Directory containing reads without adaptor and low quality bases
trimming_report: Directory containing trimming report for each sample
trimmed_reads_bases_count.txt: Tab delimited table recording number of reads and bases of raw data and trimmed data

Involved options:

--demultiplexed: Directory containing demultiplexed raw data
--trimmed: Output directory containing adaptor and low quality bases trimmed reads





trimmed_reads_bases_count.txt

Query Preparation

We need to prepare:

(1) full coding

(2) amino acid sequences of reference in fasta format.

Dependencies: Bioperl

Input:
Oreochromis_niloticus.frames.fas

Usage:

```
$ perl ./auto_assemble_pipeline/data_preparation/query_translate.pl \
--predicted_frames Oreochromis_niloticus.frames.fas \
--nucleo_out Oreochromis_niloticus.dna.fas \
--aa_out Oreochromis_niloticus.aa.fas
```

Involved options:

--predicted_frames: DNA Sequences of targeted loci with redundant nucleotides
--nucleo_out: Full coding DNA sequences of targeted loci
--aa_out: Amino acid sequences of targeted loci

Output:
Oreochromis_niloticus.dna.fas: Full coding DNA sequences of targeted loci
Oreochromis_niloticus.aa.fas: Amino acid sequences of targeted loci

Oreochromis_niloticus.frames.fas

```
>Danio_rerio.1.10018393.10018273.1
```

First nucleotide is first codon

```
>Danio rerio.1.10172087.10171964.2
```

Second nucleotide is first codon

```
>Danio_rerio.1.18292699.18292829.3
```

Third nucleotide is first codon

This file can be generated from ./auto_assemble_pipeline/query_preparation/predictFrames.

Please refer to ./auto_assemble_pipeline/query_preparation/predictFrames.README for more detail.

Oreochromis_niloticus.aa.fas

```
>Danio_rerio.1.10018393.10018273
PQTDSKVNGTALSSPSTSSQRSDSSLPLLRVAASQTTDTM
>Danio_rerio.1.10132455.10132285
VTESKLELEKSLKLSRKLRKELNGLTEWLAATDAELTRRSAVDGMPSDLKDEVAWAQ
```

Oreochromis_niloticus.dna.fas

```
>Danio_rerio.1.10018393.10018273
CCTCAGACGGATTCCAAGGTAAACGGCACAGCTCTGTCCTCCCCATCCACCTCCTCTCAGCGTTC
>Danio_rerio.1.10132455.10132285
GTGACAGAGAGTAAGTTGGAGCTGGAGAAGAGTCTGAAGTTGTCAAGGAAGCTGCGTAAGGAGCT
```

All inputs for assembly has been prepared.
Let's start assembling now.

The main script is placed under ./auto_assemble_pipeline/assemble/assemble.pl. This script calls another 6 scripts to recover assemblies.

6 scripts represent 6 steps of assembly. They are called by main script in following procedure:

1) ./auto_assemble_pipeline/assemble/rmdup.pl: Remove PCR duplicates

2) ./auto_assemble_pipeline/assemble/ubxandp.pl: Parse reads to target loci

3) ./auto_assemble_pipeline/assemble/sga_assemble.pl: Assemble reads for each locus

4) ./auto_assemble_pipeline/assemble/exonerate_best.pl: Filter unqualified contigs and find contigs which might be furtherly assembled

5) ./auto_assemble_pipeline/assemble/merge.pl: Assemble contigs further and retrieve best contigs for each locus

6) ./auto_assemble_pipeline/assemble/reblast.pl: Remove potential paralogs

Normally, we run the whole pipeline (cleaned reads in, orthologue assemblies out), which includes 3 steps

System requirements

Softwares: (Please put them under $PATH)

perl v5.18 or higher
usearch v10.0.240 or higher
sga v0.10.15 or higher
exonerate v2.2.0 or higher

Perl module:

Bio::Seq (Included in Bioperl)
Parallel::Forkmanager
Sys::Info

Check requirements of assembling

Before running the script, we need to check requirements which can be checked by "--check_depends".

Please provide "--script_path", if 6 called scripts is not placed under $PATH

Usage:

```
$ perl ./auto_assemble_pipeline/assemble/assemble.pl \
--check_depends \
--script_path ./auto_assemble_pipeline/assemble
```

Involved options:

--check_depends: Check all dependencies for assemble.pl
--script_path: Path to the scripts

```
-bash-4.1$ perl ./auto_assemble_pipeline/assemble/assemble.pl \
> --check_depends \
> --script_path ./auto_assemble_pipeline/assemble
Currently used interpreter is "/home/users/cli/bin/perl"

Version of your perl interpreter (/home/users/cli/bin/perl) is v5.24

All modules are properly installed

All softwares are properly installed

All scripts are found under "./auto_assemble_pipeline/assemble"

-bash-4.1$
```

If scripts are placed under $PATH

Since --script_path is not specified, we assume all scripts lied under \$PATH. You will use default interpreter (/usr/bin/env perl) to run the wrapper

All modules are properly installed

All softwares are properly installed

All scripts are found under $PATH

Check the existence of sequences of reference in given genome

We must ensure all sequences of reference can be found in given genome, or all sequences in this loci will be lost

Input:
Oreochromis_niloticus.dna.fas
Oreochromis_niloticus.genome.fas

Usage:

```
$ perl ./auto_assemble_pipeline/assemble/assemble.pl \
--check_query \
--queryn Oreochromis_niloticus.dna.fas \
--db Oreochromis_niloticus.genome.fas \
--dbtype nucleo \
--script_path ./auto_assemble_pipeline/assemble
```

Output:
Oreochromis_niloticus.genome.fas.udb

Involved options:

--check_query: Check query sequences (--queryn) existing in given database, and return list of missing query, then exit
--queryn: Nucleotide sequences of target loci in fasta format
--db: Path to DNA or amino acid database, either in fasta or udb format
--dbtype: Database type either 'nucleo' for DNA or 'prot' for amino acid database
—script_path: Path to the scripts

```
-bash-4.1$ perl ./auto_assemble_pipeline/assemble/assemble.pl \
> --check_query \
> --queryn Oreochromis_niloticus.dna.fas \
> --db Oreochromis_niloticus.genome.fas \
> --dbtype nucleo \
> --script_path ./auto_assemble_pipeline/assemble
Number of avaliable CPU is 32

Wrapper will be run in 32 threads

You will use '/home/users/cli/bin/perl' to run the scripts under ./auto_assemble_pipeline/assemble

00:00 44Mb    100.0% Reading Oreochromis_niloticus.genome.fas
00:00 12Mb    100.0% Word stats
00:00 12Mb    100.0% Alloc rows
00:00 17Mb    100.0% Build index
00:00 17Mb    100.0% Rows
00:00 17Mb   Buffers (17675 seqs)
00:00 34Mb    100.0% Seqs
00:00 37Mb   CPU has 32 cores, defaulting to 10 threads
00:05 141Mb   100.0% Masking
00:00 9.3Mb   100.0% Reading rows
00:00 9.5Mb  Reading pointers...done.
00:00 10Mb   Reading db seqs...done.
00:02 153Mb   100.0% Searching, 100.0% matched
#### All genes are found in provided database ####
-bash-4.1$
```

Requirements and existence of target loci in given genome have been checked. Let's start assemble.

Assemble

Input of assemble including:

(1) trimmed

(2) Oreochromis_niloticus.aa.fas

(3) Oreochromis_niloticus.dna.fas

(4) Oreochromis_niloticus.genome.fas

Usage:

```
$ perl ./auto_assemble_pipeline/assemble/assemble.pl \
--trimmed trimmed \
--queryp Oreochromis_niloticus.aa.fas \
--queryn Oreochromis_niloticus.dna.fas \
--db Oreochromis_niloticus.genome.fas \
--dbtype nucleo \
--ref_name Oreochromis_niloticus \
--outdir assemble_result \
--script_path ./auto_assemble_pipeline/assemble
```

Involved options:

--trimmed: Directory containing reads without adaptor and low quality bases
--queryp: Amino acid sequences of target loci in fasta format
--queryn: Nucleotide sequences of target loci in fasta format
--db: Path to DNA or amino acid database, either in fasta or udb format
--dbtype: Database type either 'nucleo' for DNA or 'prot' for amino acid database
--ref_name: Substitute name of target loci as --ref_name in the output of last step (reblast.pl), disabled in default
--outdir: Directory to pipeline output
--script_path: Path to the scripts

Several folders and files will be generated during the execution:

1) run_dir: Folder generated in step 1. All intermediate outputs will be generated under this folder.

2) samplelist.txt: File generated in step 1. A list includes the name of all sample

3) rmdup_reads_bases_count.txt: File generated in step 1. A tab delimited table records number of reads and bases before and after removing PCR duplicates

4) enriched_loci.txt: File generated after step 6. A tab delimited table records number of total loci, number of enriched loci and percentage of enriched loci for each sample

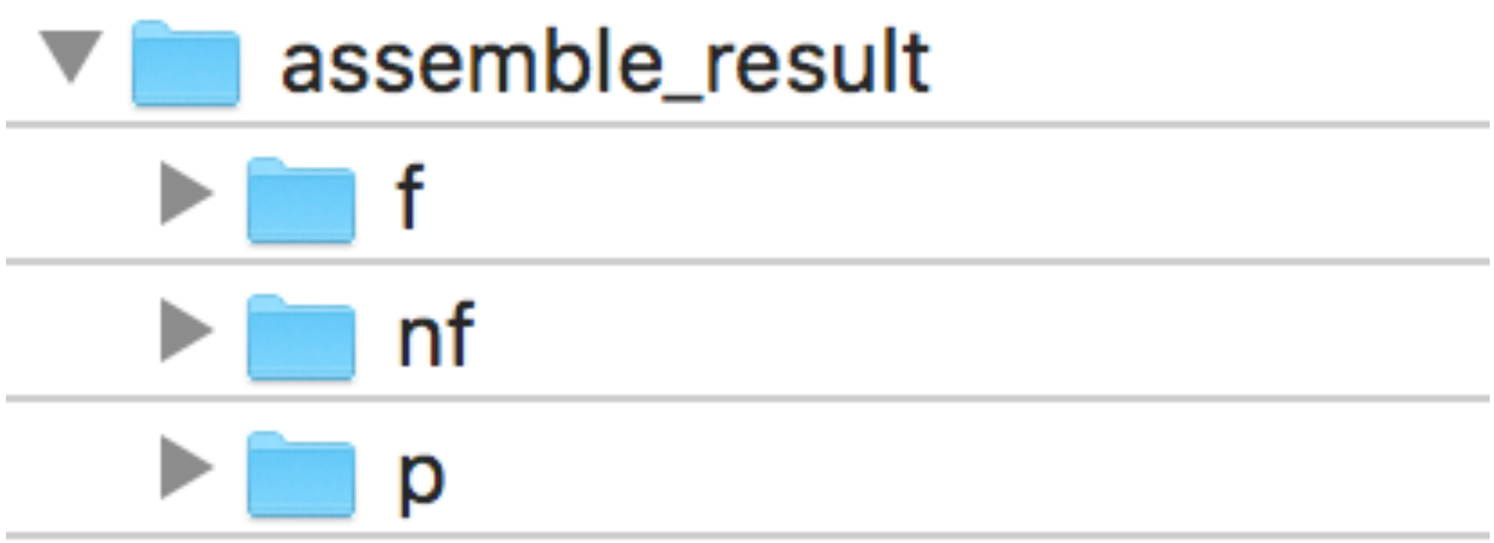
5) Oreochromis_niloticus.genome.fas.udb: File generated in step 6. udb of "Oreochromis_niloticus.genome.fas".

Output will be placed under "assemble_result" including 3 folders:

1) nf: folder containing full coding nucleotide sequences

2) f: folder containing coding sequences with flankings

3) p: folder containing amino acid sequences



If something goes wrong at intermediate step, don't worry, assemble.pl is able to restart from intermediate step. It can also stop at the step you want

Further processing

After assembling, recovered assemblies need to be further processed before being fed into downstream analysis. Further processing includes:

1) Adding or deleting sequences from datasets

2) Aligning

3) Filtering

4) Summary statistics

Adding to or deleting sequences from datasets

Sequences must be added or deleted before aligning

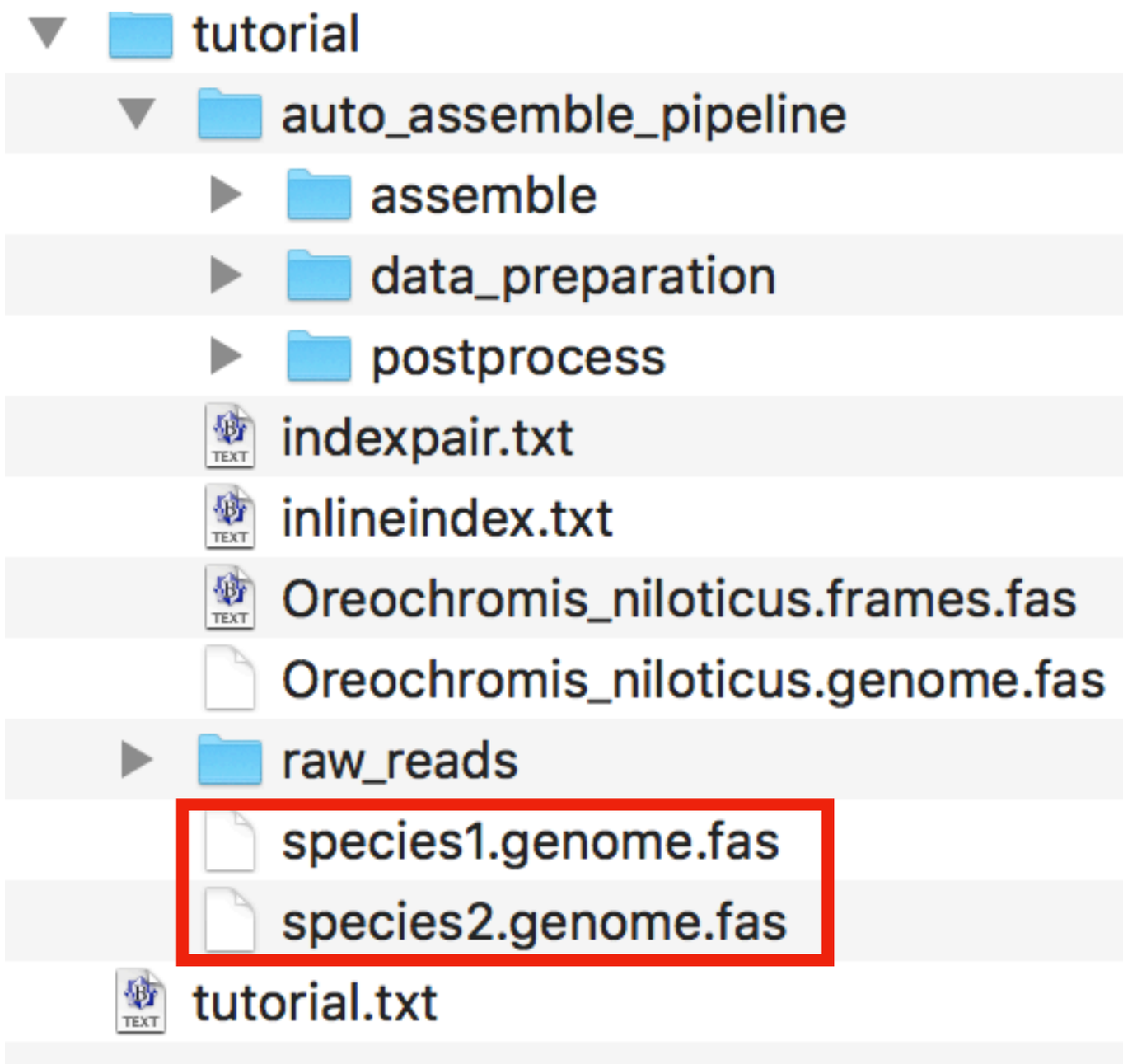
Add orthologue sequences

Extract orthologue sequences from existing genomes

Add them into datasets

Extract orthologue sequences from existing genomes

We extract sequences orthology to loci in "Oreochromis_niloticus.dna.fas" from "species1.genome.fas" and “species2.genome.fas"



Dependencies:
usearch v10.0.240 or higher
BioPerl v1.007001 or higher

Input:
Oreochromis_niloticus.dna.fas
Oreochromis_niloticus.genome.fas
species1.genome.fas
species2.genome.fas

Usage:

```
$ perl auto_assemble_pipeline/postprocess/get_orthologues.pl \
--query Oreochromis_niloticus.dna.fas \
--querydb Oreochromis_niloticus.genome.fas \
--subdb "species1.genome.fas species2.genome.fas" \
--subname "species1 species2" \
--outdir orthologues \
--cpu 12
```

Involved options:

—query: File contains full coding nucleotide sequences only
—querydb: Space delimited list of one or more nucleotide databases belonging to the same query species in either fasta (masked is better) or udb format
—subdb: Space delimited list of nucleotide databases of subjects in fasta format (masked is better), single database for each species
—subname: Space delimited list of subject name in output, which is one-to-one match to the list of subject databases. If this option is not specifed, the name of corresponding sequences will be the prefix of database file.
—outdir Name of output directory, which has 2 subfolders including "nf" for coding sequences and "p" for amino acid sequences. DO NOT NAME OUTDIR AS "qblasts", “qblastsout", "reblast_input.fas" or "reblast" which are names of intermediate files
—cpu Limit the number of CPUs, 1 in default

Output:
orthologues: Directory includes sequences orthology to target loci

Oreochromis_niloticus.genome.fas.udb: udb of "Oreochromis_niloticus.genome.fas".

Add orthologues into datasets

Dependencies: no

Input
./assemble_result/nf
./orthologues/nf

Usage:

```
$ perl auto_assemble_pipeline/postprocess/merge_loci.pl \
--indir "./assemble_result/nf ./orthologues/nf" \
--outdir merged_nf \
--min_seq 3
```

Involved options:

--indir
List of dir containing sequences
--outdir
Dir containing merged loci files
--min_seq
Minimum sequences required in merged file, 2 in default

Output:
merged_nf: Dir containing merged loci files

Merged sequences

```
>Oreochromis_niloticus
ATGAATTTCTTGCGTAATCGTCTCGTTGTTCTGGGCCTGGTGTTGTTGGCCACATTACTGCTCTACCTTCTGCTGCCATCCATTCGCCAG
>test1
CGTCTTGTTGTCCTGGGCCTGGTGTTGTTGGCCACATTACTGCTGTACCTGCTGCTGCCGTCCATTCGCCAGGGCAGCATGGAGCCGTCC
>species1
ATGAATTTCTTGCGTAATCGTCTCGTTGTTCTGGGCCTGGTGTTGTTGGCCACATTACTGCTCTACCTTCTGCTGCCATCCATTCGCCAG
>species2
ATGAATTTCTTGCGTAATCGTCTCGTTGTTCTGGGCCTGGTGTTGTTGGCCACATTACTGCTCTACCTTCTGCTGCCATCCATTCGCCAG
```

Delete unneeded sequences

Dependencies: Nothing

Input:
merged_nf

Usage:

```
$ perl auto_assemble_pipeline/postprocess/pick_taxa.pl \
--indir merged_nf \
--outdir merged_nf_deselected \
--deselected_taxa "species2"
```

Output:
merged_nf_deselected: Dir containing sequences of without discarded taxon

Involved options:

--indir
Dir containing unaligned sequences
--outdir
Dir containing sequences of selected taxon
--deselected_taxa
List of taxa want to be discarded, each taxon is delimited by space

Same locus but “species2” is discarded

```
>Oreochromis_niloticus
ATGAATTTCTTGCGTAATCGTCTCGTTGTTCTGGGCCTGGTGTTGTTGGCCACATTACTGCTCTACCTTCTGCTGCCATCCATTCGCCA
>test1
CGTCTTGTTGTCCTGGGCCTGGTGTTGTTGGCCACATTACTGCTGTACCTGCTGCTGCCGTCCATTCGCCAGGGCAGCATGGAGCCGTC
>species1
ATGAATTTCTTGCGTAATCGTCTCGTTGTTCTGGGCCTGGTGTTGTTGGCCACATTACTGCTCTACCTTCTGCTGCCATCCATTCGCCA
```

Aligning

Dependencies:
(1) BioPerl v1.007001 or higher
(2) Mafft v7.294b or higher

Input:
merged_nf

Usage:

```
$ perl auto_assemble_pipeline/postprocess/mafft_aln.pl \
--dna_unaligned merged_nf \
--dna_aligned merged_nf_aligned \
--cpu 12
```

Output:
merged_nf_aligned: Dir containing nucleotide sequences aligned in codon

Involved options:

--dna_unaligned: Dir containing unaligned nucleotide sequences
--dna_aligned: Dir containing aligned nucleotide sequences, named as "xx_aligned" if this option is not specified
—non_codon_aln: Do not align nucleotide sequences in codon. This option is turned off by default
—cpu: Limit the number of CPUs, 1 in default.

Aligned sequences

```
>Oreochromis_niloticus
ATGAATTTCTTGCGTAATCGTCTCGTTGTTCTGGGCCTGGTGTTGTTGGCCACATTACTGCTCTACCTTCTGCTGCCATCCATTCGCCAG
>test1
------------------CGTCTTGTTGTCCTGGGCCTGGTGTTGTTGGCCACATTACTGCTGTACCTGCTGCTGCCGTCCATTCGCCAG
>species1
ATGAATTTCTTGCGTAATCGTCTCGTTGTTCTGGGCCTGGTGTTGTTGGCCACATTACTGCTCTACCTTCTGCTGCCATCCATTCGCCAG
>species2
ATGAATTTCTTGCGTAATCGTCTCGTTGTTCTGGGCCTGGTGTTGTTGGCCACATTACTGCTCTACCTTCTGCTGCCATCCATTCGCCAG
```

Filtering

Some sequences may result in poorly alignment, we need to remove them

Dependencies:
(1) BioPerl v1.007001 or higher
(2) Mafft v7.294b or higher

Input:
filter_test

Usage:

```
$ perl auto_assemble_pipeline/postprocess/filter.pl \
--indir filter_test \
--filtered filtered \
--cpu 12
```

Output:
merged_nf_filtered

Involved options:
--indir: Dir containing unfiltered alignments
--filtered: Dir containing filtered alignments
--cpu: Limit the number of CPUs, 1 in default

Summary statistics

Dependencies:
Bio::AlignIO (included in Bioperl)
Bio::Align::DNAStatistics (included in Bioperl)

Input:
merged_nf_aligned
./assemble_result/f

Usage:

```
$ perl auto_assemble_pipeline/postprocess/statistics.pl \
--nf_aligned merged_nf_aligned \
--f_unaligned ./assemble_result/f
```

Involved options:
--nf_aligned:
Folder comprising aligned full-coding sequences
--f_unaligned:
Folder comprising unaligned whole sequences (include flanking sequences)

Output:
1) loci_summary.txt: Tab delimited table of summary statistics for each locus
2) sample_summary.txt: Tab delimited table of summary statistics for each sample

Thanks