De novo sequencing and Assembly

Andreas Gisel
International Institute of Tropical Agriculture (IITA)
Ibadan, Nigeria

The Principle of Mapping

reads

```
good, ood_, d_mo, morn, orni, ning, ing_,
g_be, beau, auti, utif, iful, ul_w, _wor orld
```

reference

```
good_morning_beautiful_world
```

mapping

```
          ing_    utif
   d_mo ning    auti    _wor
 ood_ orni    beau    ul_w
good morn   g_be    iful   orld

good_morning_beautiful_world
```

consensus

```
good_morning_beautiful_world
```

The Principle of Assembly

reads

```
good, ood_, d_mo, morn, orni, ning, ing_,
g_be, beau, auti, utif, iful, ul_w, _wor orld
```

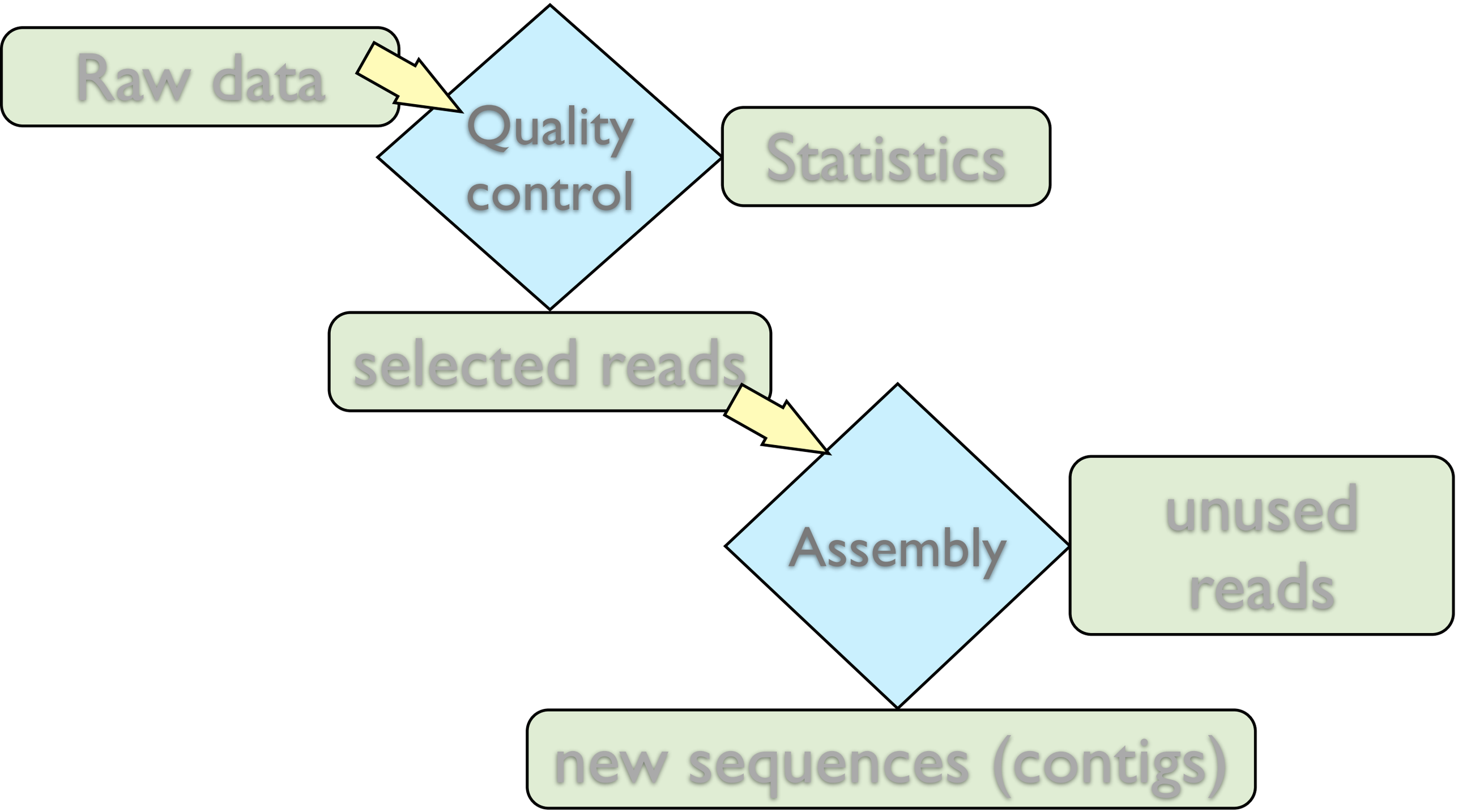
assembly

```
good
 ood_
   d_mo
    morn
     orni
       ning
        ing_
          g_be
            beau
              auti
               utif
                 iful
                   ul_w
                     _wor
                       orld
```

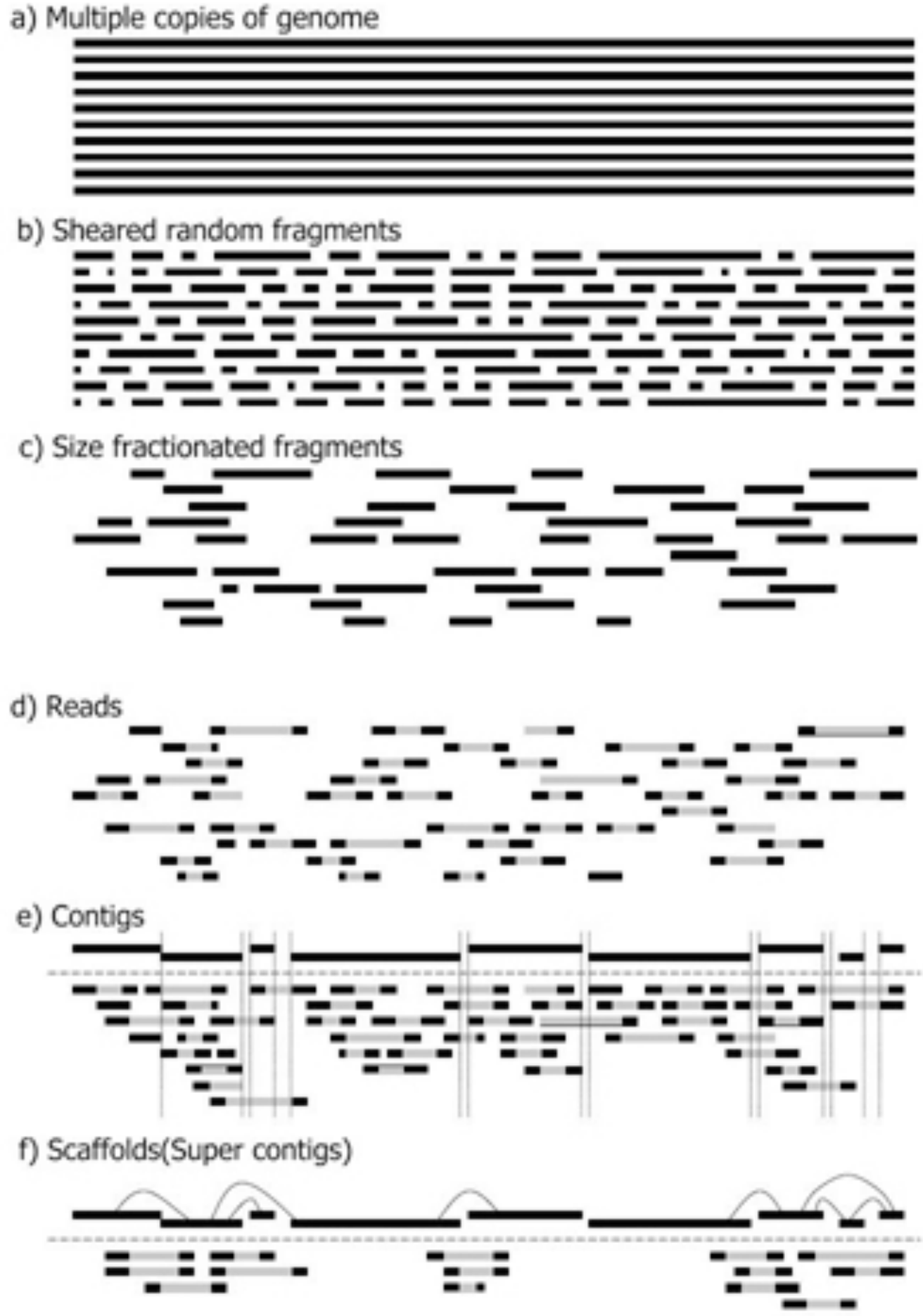
consensus

```
good_morning_beautiful_world
```

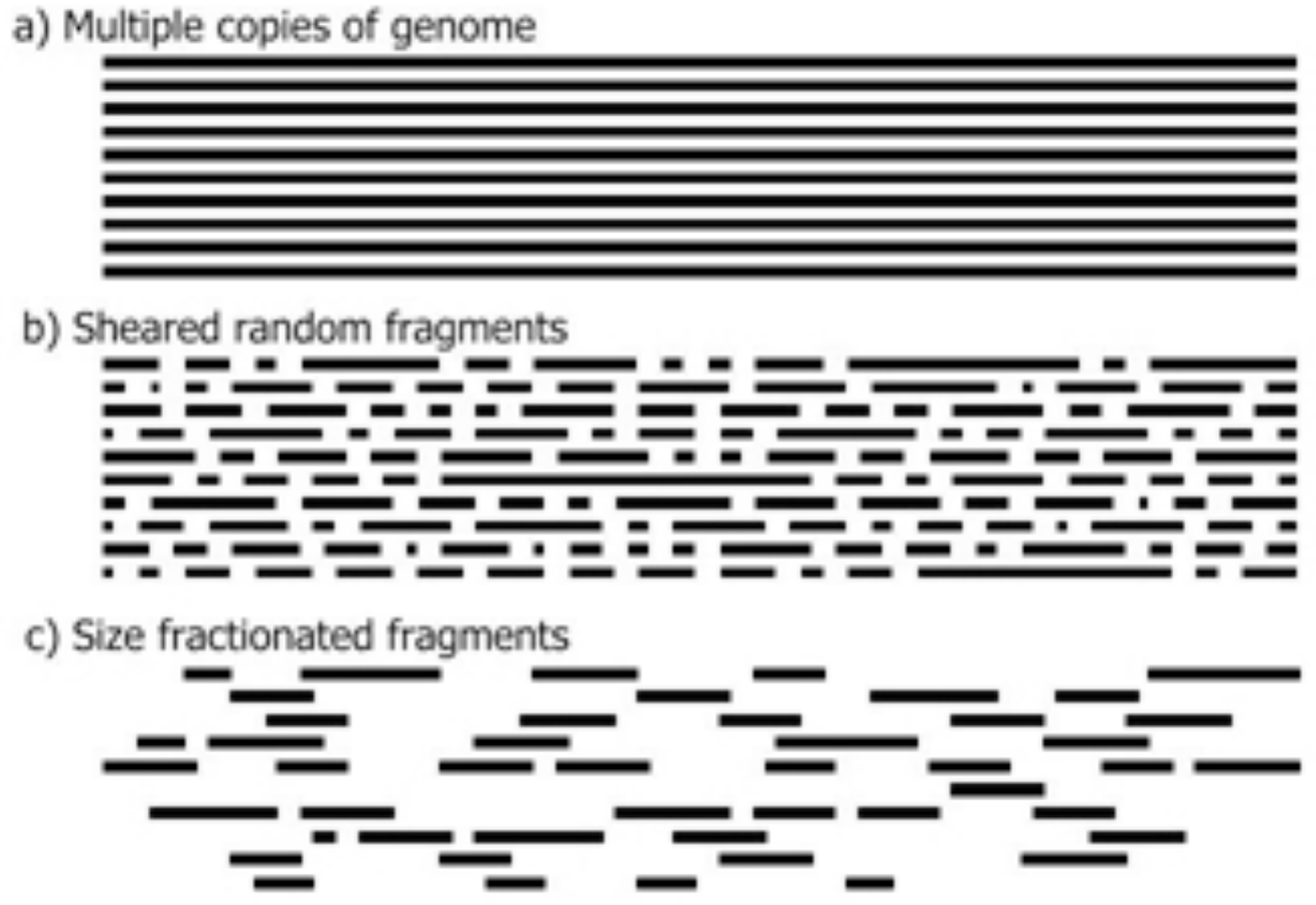
Workflow for Assembly

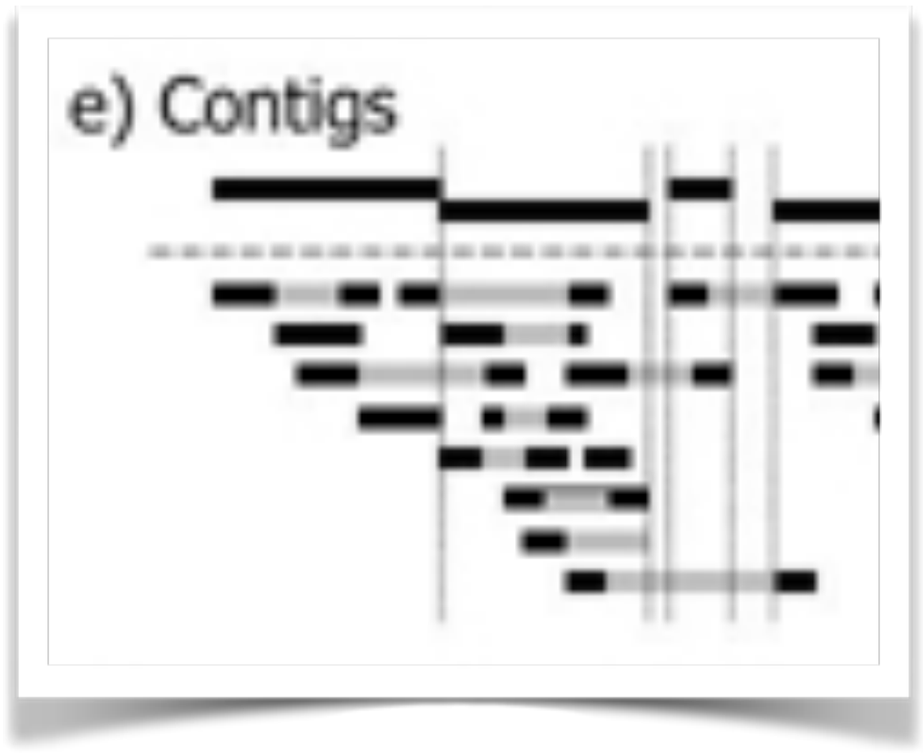


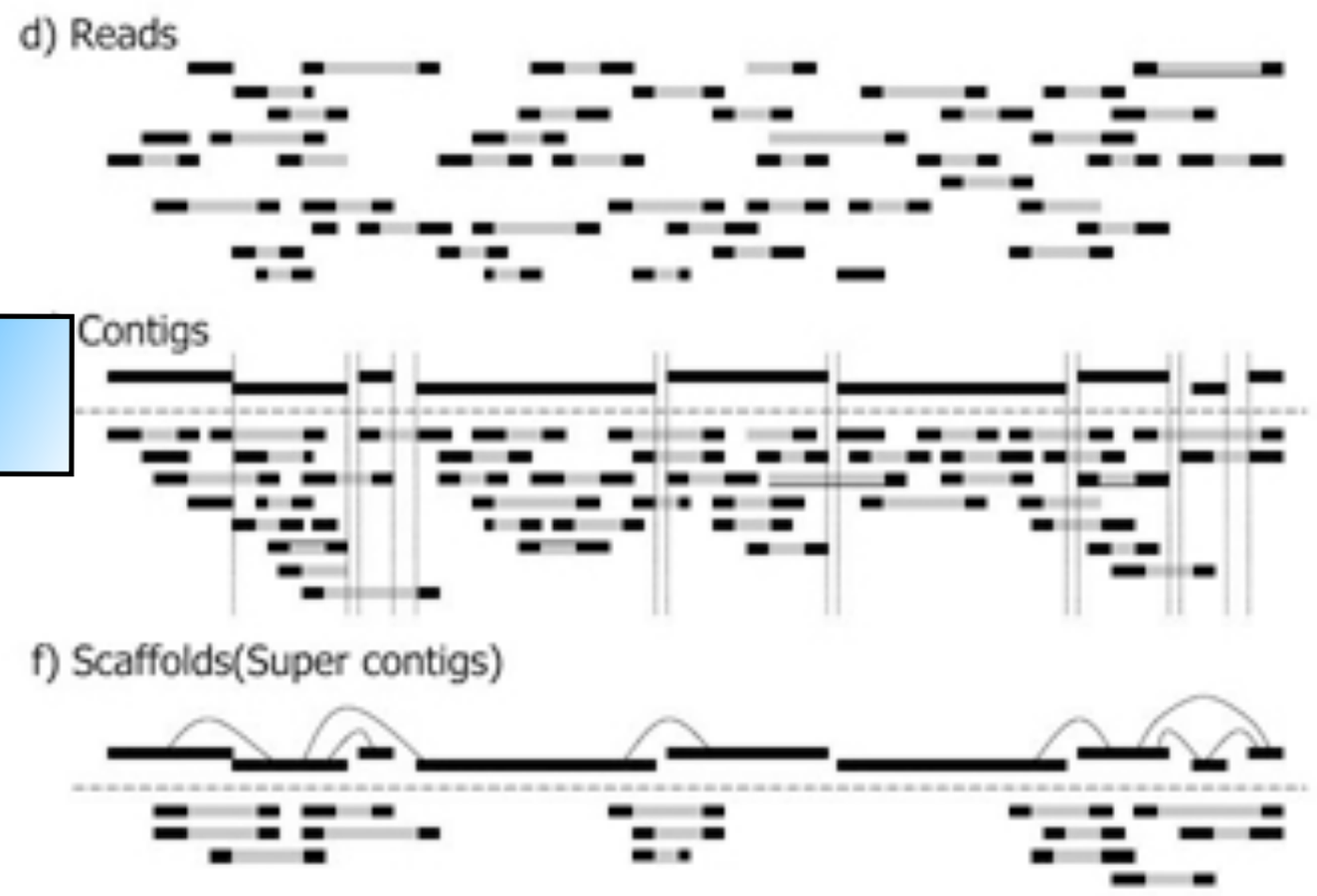
Workflow


a) Multiple copies of genome
b) Sheared random fragments
c) Size fractionated fragments
d) Reads
e) Contigs
f) Scaffolds(Super contigs)

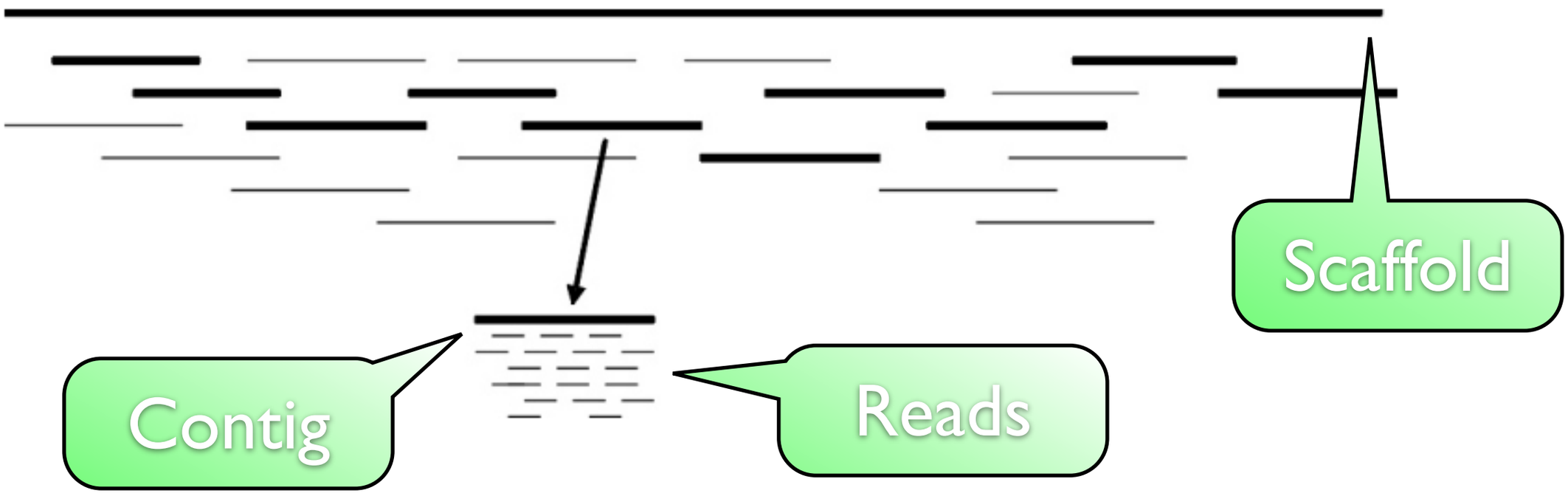
Workflow


a) Multiple copies of genome
b) Sheared random fragments
c) Size fractionated fragments


e) Contigs


d) Reads
Contigs
f) Scaffolds(Super contigs)

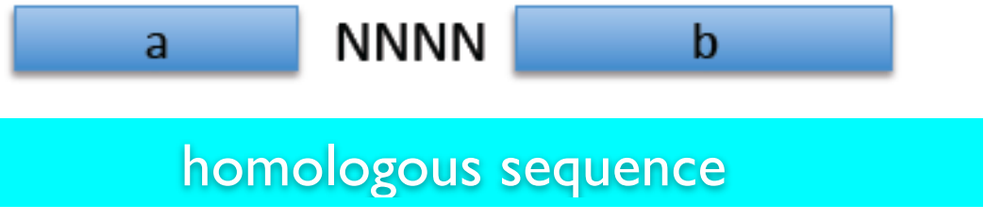
Contigs - Scaffolds



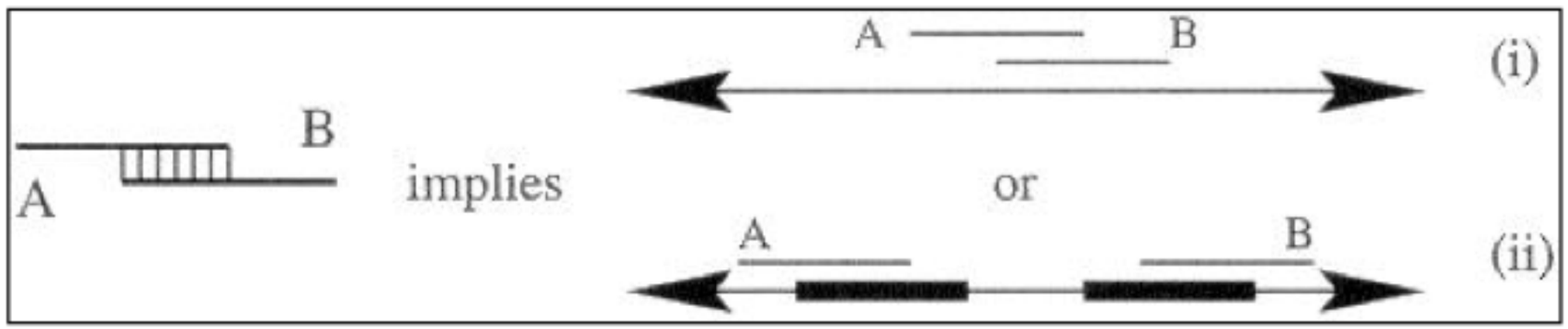
Contigs - Scaffolds

Connect Contigs with:

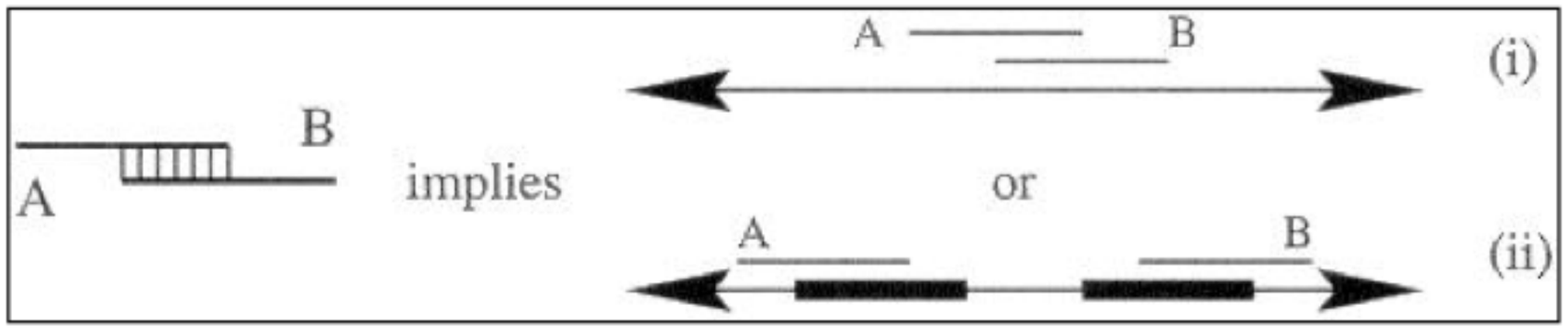
- ☑ mate-pair information
- ☑ homology data
- ☑ physical maps
- ☑ gene synteny

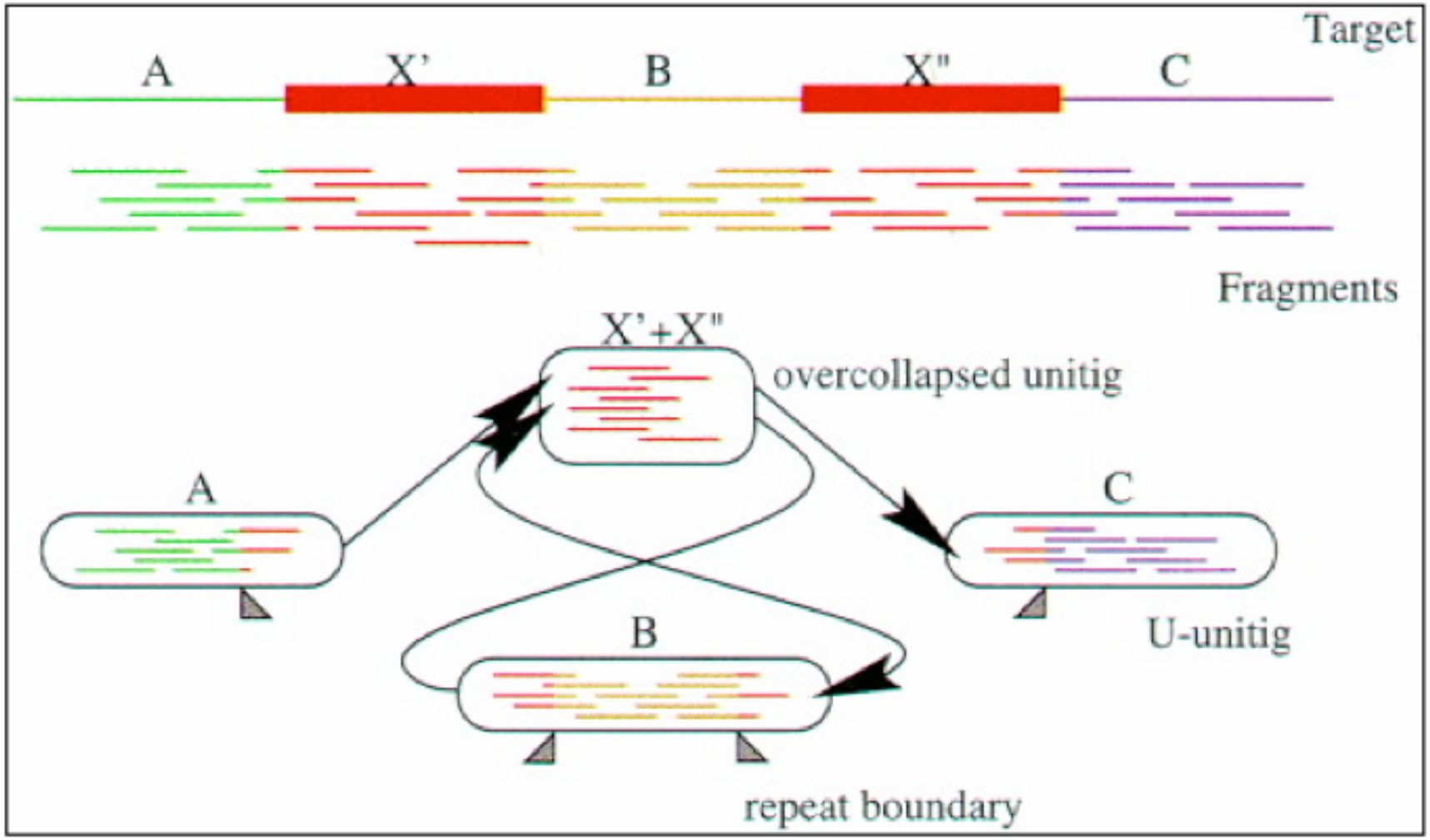


Problem of Repeats



Problem of Repeats



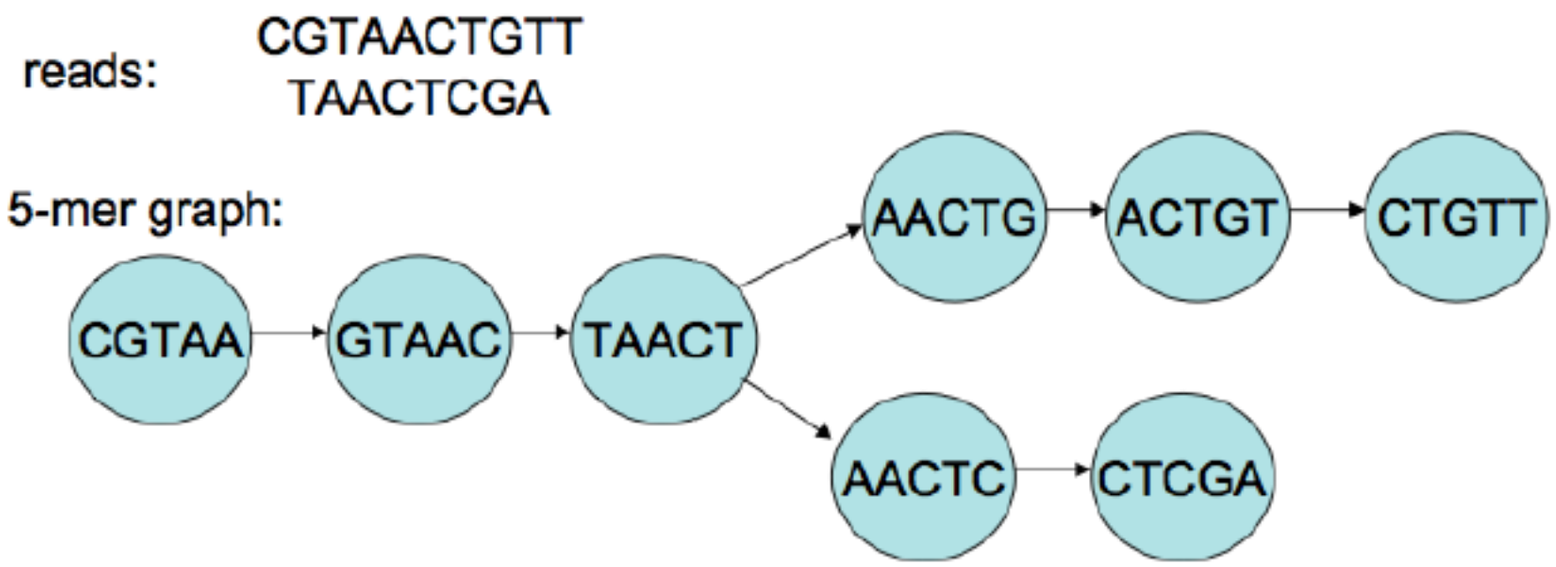


deBruijn graph

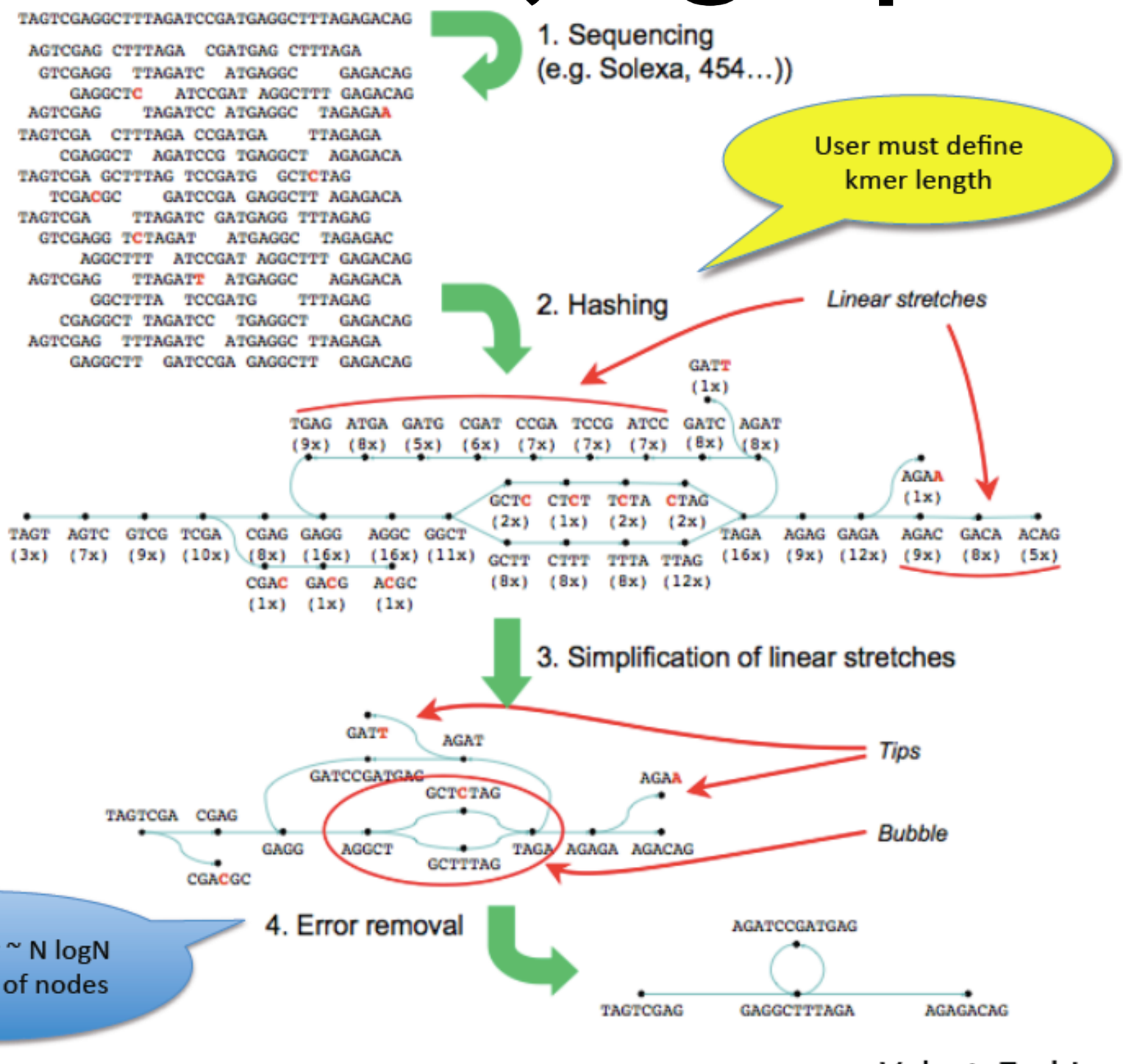
Nodes are k-mers and not reads

☑ small k-mers dense graph (not good)

☑ large k-mers sparse graph (good, results in larger contigs, but need more reads)



deBruijn graph



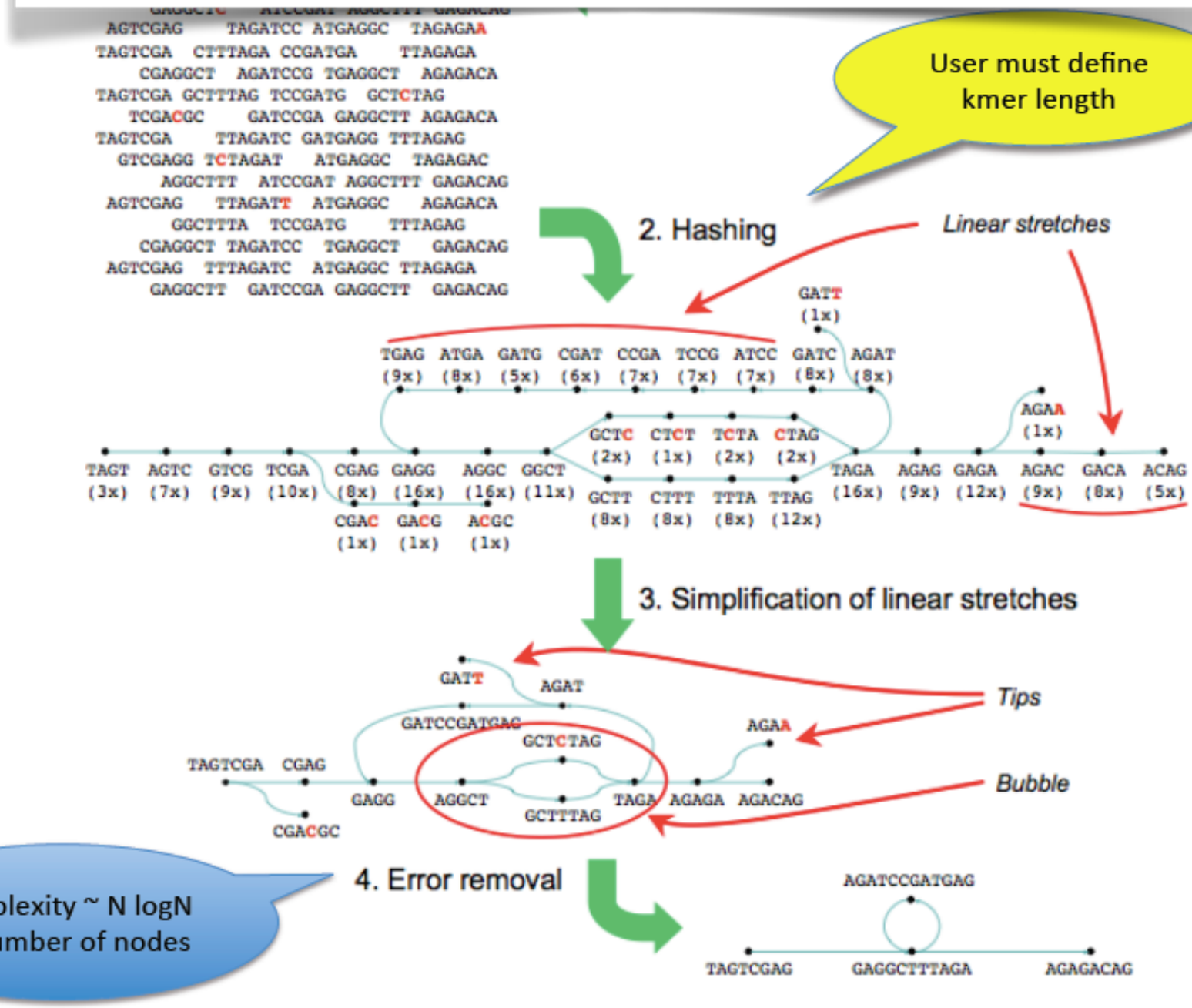
Velvet: Zerbino & Birney

TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG

TAGTCGAG

GAGGCTTTAGA AGAGACAG

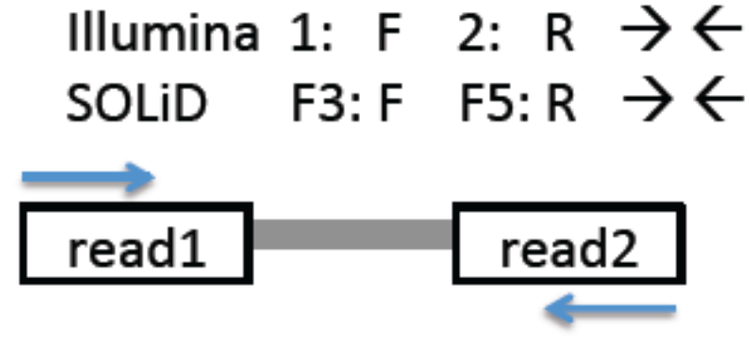
AGATCCGATGAG



Velvet: Zerbino & Birney

Data

- Pair-end
  - 200bp
  - 600bp
  - 800bp
- Mate pair
  - 3Kb
  - 8Kb
  - 20Kb



Assembly measures

- ☑ Sum of Contig length
  - Theoretical genome size
- ☑ Number of contigs
- ☑ N50
  - Contig or scaffold N50 is a weighted median statistic such that 50% of the entire assembly is contained in contigs or scaffolds equal to or larger than this value
- ☑ Accuracy

Assembly measures

Grapevine clone: 6 lanes ($100bp$), insert size 200 ± 50
Coverage: $89\times$

	AbySS	SOAPdenovo	CLC
# Scaf num	289,854 (244k)	127,648 (368k)	151,288 (423k)
Tot Scaf. length (bp)	562M (158M)	257M (285M)	339M (382M)
Max Scaf length (bp)	89,700 (12k)	59,054 (36k)	69,474 (70k)
Mean Scaf lgth (bp)	1942 (649)	2014 (776)	2241 (904)
N50 length	2634 (872)	3186 (2038)	3328 (1823)
time	18h 49m (12h)	8h 57m (1d)	6h 45m (7h)
RAM available (GB)	130 (240)	240 (120)	120 (120)
RAM used (GB)	$\sim$ 90 (102)	143 (70)	$\sim$ 80 (60)
CPUs	80 (80)	8 (8)	8 (8)

Assemblers

- ☑ Phrap
- ☑ CAP3
- ☑ Celera assembler
- ☑ CABOG (modified Celera assembler for 454)
- ☑ Newbler
- ☑ Arachne
- ☑ AMOS (A Modular Open-.−Source whole genome assembler)
- ☑ ABBA (Assembly Boosted by Amino Acid Sequences)
- ☑ MIRA
- ☑ ABySS
- ☑ Euler
- ☑ Velvet
- ☑ SOAPdenovo
- ☑ ALLPATHS, ALLPATHS-.−LG

Assembler

☑ Velvet

- http://www.ebi.ac.uk/~zerbino/velvet/

☑ ABySS

- http://www.bcgsc.ca/platform/bioinfo/software/abyss/

☑ SOAPdenovo

- http://soap.genomics.org.cn/soapdenovo.html

Velvet

EMBL-EBI

http://www.ebi.ac.uk/~zerbino/velvet/

Velvet

Sequence assembler for very short reads

- **Current version: 1.2.08**
- **Manual and extension for Columbus in pdf format**
- **Public Git URL: git clone git://github.com/dzerbino/velvet.git**
- **For up-to-date info, you can consult and/or subscribe to the mailing list.**
- **For transcriptomic assembly Velvet is extended by Oases.**

News

29/03/2011: Velvet 1.1

Velvet is now multithreaded, thanks to the use of the OMP library.

Velvet

velveth

velveth helps you construct the dataset for the following program, velvetg, and indicate to the system what each sequence file represents

velvetg

velvetg is the core of Velvet where the de Bruijn graph is built then manipulated.

Velvet

velveth

velveth -h

Usage:
./velveth directory hash_length {[-file_format][-read_type] filename1 [filename2 ...]} {...} [options]

directory : directory name for output files
hash_length : EITHER an odd integer (if even, it will be decremented) <= 31 (if above, will be reduced)
: OR: m,M,s where m and M are odd integers (if not, they will be decremented) with m < M <= 31 (if above, will be reduced)
and s is a step (even number). Velvet will then hash from k=m to k=M with a step of s
filename : path to sequence file or - for standard input

File format options:
-fasta -fastq -raw -fasta.gz -fastq.gz -raw.gz -sam -bam

Read type options:
-short -shortPaired
-short2 -shortPaired2
-long -longPaired
-reference

Options:
-strand_specific : for strand specific transcriptome sequencing data (default: off)
-reuse_Sequences : reuse Sequences file (or link) already in directory (no need to provide original filenames in this case (default: off)
-noHash : simply prepare Sequences file, do not hash reads or prepare Roadmaps file (default: off)
-create_binary : create binary CnyUnifiedSeq file (default: off)

Velvet

velveth

velveth -h

Synopsis:

- Short single end reads:
    velveth Assem 29 -short -fastq s_1_sequence.txt

- Paired-end short reads (remember to interleave paired reads):
    velveth Assem 31 -shortPaired -fasta interleaved.fna

- Two channels and some long reads:
    velveth Assem 43 -short -fastq unmapped.fna -longPaired -fasta SangerReads.fasta

- Three channels:
    velveth Assem 35 -shortPaired -fasta pe_lib1.fasta -shortPaired2 pe_lib2.fasta -short3 se_lib1.fa

Output:
    directory/Roadmaps
    directory/Sequences
        [Both files are picked up by graph, so please leave them there]

Velvet

velvetg

velvetg

Usage:
./velvetg directory [options]

directory : working directory name

Standard options:
-cov_cutoff <floating-point|auto> : removal of low coverage nodes AFTER tour bus or allow the system to infer it
(default: no removal)
-ins_length <integer> : expected distance between two paired end reads (default: no read pairing)
-read_trkg <yes|no> : tracking of short read positions in assembly (default: no tracking)
-min_contig_lgth <integer> : minimum contig length exported to contigs.fa file (default: hash length * 2)
-amos_file <yes|no> : export assembly to AMOS file (default: no export)
-exp_cov <floating point|auto> : expected coverage of unique regions or allow the system to infer it
(default: no long or paired-end read resolution)
-long_cov_cutoff <floating-point>: removal of nodes with low long-read coverage AFTER tour bus
(default: no removal)

velvetg

velvetg

Velvet

Advanced options:
-ins_length2 <integer> : expected distance between two paired-end reads in the second short-read dataset (default: no read pairing)
-ins_length_long <integer> : expected distance between two long paired-end reads (default: no read pairing)
-ins_length*_sd <integer> : est. standard deviation of respective dataset (default: 10% of corresponding length)
[replace '*' by nothing, '2' or '_long' as necessary]
-scaffolding <yes|no> : scaffolding of contigs used paired end information (default: on)
-max_branch_length <integer> : maximum length in base pair of bubble (default: 100)
-max_divergence <floating-point>: maximum divergence rate between two branches in a bubble (default: 0.2)
-max_gap_count <integer> : maximum number of gaps allowed in the alignment of the two branches of a bubble (default: 3)
-min_pair_count <integer> : minimum number of paired end connections to justify the scaffolding of two long contigs (default: 5)
-max_coverage <floating point> : removal of high coverage nodes AFTER tour bus (default: no removal)
-coverage_mask <int> : minimum coverage required for confident regions of contigs (default: 1)
-long_mult_cutoff <int> : minimum number of long reads required to merge contigs (default: 2)
-unused_reads <yes|no> : export unused reads in UnusedReads.fa file (default: no)
-alignments <yes|no> : export a summary of contig alignment to the reference sequences (default: no)
-exportFiltered <yes|no> : export the long nodes which were eliminated by the coverage filters (default: no)
-clean <yes|no> : remove all the intermediary files which are useless for recalculation (default : no)
-very_clean <yes|no> : remove all the intermediary files (no recalculation possible) (default: no)
-paired_exp_fraction <double>: remove all the paired end connections which less than the specified fraction of the expected count (default: 0.1)
-shortMatePaired* <yes|no> : for mate-pair libraries, indicate that the library might be contaminated with paired-end reads (default no)
-conserveLong <yes|no> : preserve sequences with long reads in them (default no)

Output:
directory/contigs.fa : fasta file of contigs longer than twice hash length
directory/stats.txt : stats file (tab-spaced) useful for determining appropriate coverage cutoff
directory/LastGraph : special formatted file with all the information on the final graph
directory/velvet_asm.afg : (if requested) AMOS compatible assembly file

Velvet

velvetg

velvetg

```
andreas@popeye:~/circle2/MAC18-17d/ciRNAse/data/kmer_19$ ll
total 997664
drwxr-xr-x 2 andreas andreas      4096 Feb 16  2012 ./
drwxr-xr-x 5 andreas andreas      4096 Jul 10  2012 ../
-rw-r--r-- 1 andreas andreas     64642 Feb 16  2012 contigs.fa
-rw-r--r-- 1 andreas andreas  14998656 Feb 16  2012 Graph2
-rw-r--r-- 1 andreas andreas  14998656 Feb 16  2012 LastGraph
-rw-r--r-- 1 andreas andreas       320 Feb 16  2012 Log
-rw-r--r-- 1 andreas andreas   2804871 Feb 16  2012 PreGraph
-rw-r--r-- 1 andreas andreas 144799894 Feb 16  2012 Roadmaps
-rw-r--r-- 1 andreas andreas 301359612 Feb 16  2012 Sequences
-rw-r--r-- 1 andreas andreas     87634 Feb 16  2012 stats.txt
-rw-r--r-- 1 andreas andreas 173490832 Feb 16  2012 UnusedReads.fa
-rw-r--r-- 1 andreas andreas 368975182 Feb 16  2012 velvet_asm.afg
```

ABySS

BC Cancer Agency
CARE + RESEARCH
An agency of the Provincial Health Service

CANADA'S MICHAEL SMITH

Log in

http://www.bcgsc.ca/platform/bioinfo/software/abyss/

Home | Platforms | Projects | Data | Training | Services | Faculty | Careers | About

You are here: Home › Platforms › Bioinformatics › GSC Software Centre › ABySS

- Platforms
  - Bioinformatics
    - Bioinformatics Licenses
    - GSC Software Centre
      - PASsiT
      - Adapter Trimming for Small RNA Sequencing
      - Spark
      - TASR
      - XpressAlign: FPGA Short Read Aligner
      - Barnacle
      - Satellog
      - HLAminer
      - Anchor

ABySS

Assembly By Short Sequences - a de novo, parallel, paired-end sequence assembler

Current release

ABySS 1.3.4

Released May 30, 2012

This release eliminates two sources of misassemblies, one in the path extension logic of SimpleGraph. Two, the default value of m, which is the minimum overlap required between two contigs to merge them, is increased from 30 to 50. This release also fixes various portability issues. A new script, abyss-fatoagp, is included to create an AGP file for GenBank submission.

More about this release…

Get ABySS for all platforms (626 kB)

Source

Get ABySS for Linux (1.7 MB)

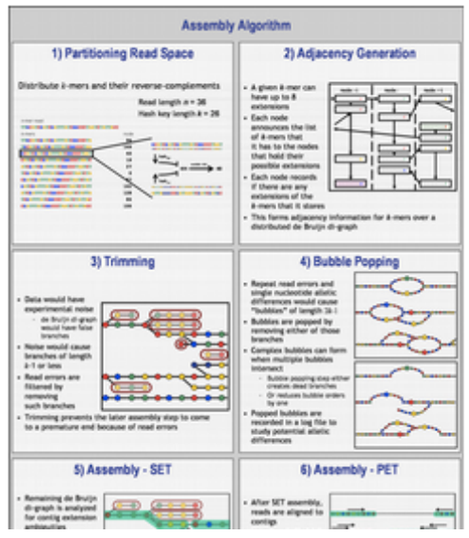
Debian package (amd64)

Project Resources

- Releases
- Documentation
- Support
- Contact address

Project owner: Shaun Jackman

Subscribe to updates for this project



ABySS

abyss-pe uses the following programs, which must be found in your PATH:

- ABYSS: de Bruijn graph assembler
- ABYSS-P: parallel (MPI) de Bruijn graph assembler
- AdjList: find overlapping sequences
- DistanceEst: estimate the distance between sequences
- MergeContigs: merge sequences
- MergePaths: merge overlapping paths
- Overlap: find overlapping sequences using paired-end reads
- PathConsensus: find a consensus sequence of ambiguous paths
- PathOverlap: find overlapping paths
- PopBubbles: remove bubbles from the sequence overlap graph
- SimpleGraph: find paths through the overlap graph
- abyss-fac: calculate assembly contiguity statistics
- abyss-filtergraph: remove shim contigs from the overlap graph
- abyss-fixmate: fill the paired-end fields of SAM alignments
- abyss-map: map reads to a reference sequence
- abyss-scaffold: scaffold contigs using distance estimates
- abyss-todot: convert graph formats and merge graphs

ABySS

Parameters of the driver script, abyss-pe

- a: maximum number of branches of a bubble [2]
- b: maximum length of a bubble (bp) [10000]
- c: minimum mean k-mer coverage of a unitig [sqrt(median)]
- d: allowable error of a distance estimate (bp) [6]
- e: minimum erosion k-mer coverage [sqrt(median)]
- E: minimum erosion k-mer coverage per strand [1]
- j: number of threads [2]
- k: size of k-mer (bp)
- l: minimum alignment length of a read (bp) [k]
- m: minimum overlap of two unitigs (bp) [30]
- n: minimum number of pairs required for building contigs [10]
- N: minimum number of pairs required for building scaffolds [n]
- p: minimum sequence identity of a bubble [0.9]
- q: minimum base quality [3]
- s: minimum unitig size required for building contigs (bp) [200]
- S: minimum contig size required for building scaffolds (bp) [s]
- t: minimum tip size (bp) [2k]
- v: use v=-v to enable verbose logging [disabled]

ABySS

☑ Assembling a paired-end library

- abyss-pe
  - name=ecoli
  - k=64
  - in='reads1.fa reads2.fa'

☑ Assembling multiple libraries

- abyss-pe
  - k=64
  - name=ecoli
  - lib='pe200 pe500'
  - pe200='pe200_1.fa pe200_2.fa'
  - pe500='pe500_1.fa pe500_2.fa'
  - se='se1.fa se2.fa'

ABySS

- ☑ Scaffolding
  - abyss-pe
    - k=64
    - name=ecoli
    - lib='pe1 pe2'
    - mp='mp1 mp2'
    - pe1='pe1_1.fa pe1_2.fa'
    - pe2='pe2_1.fa pe2_2.fa'
    - mp1='mp1_1.fa mp1_2.fa'
    - mp2='mp2_1.fa mp2_2.fa

Mate-pair are only used for scaffolding and DOES NOT contribute to the consensus

SOAPdenovo

SOAP

Short Oligonucleotide Analysis Package

SOAP-popIndel | SOAPdenovo-Trans | SOAPdenovo | About

Home | SOAPindel | SOAPsnv | SOAPsv | SOAPfusion | SOAP3 | SOAPfuse | SOAPaligner | SOAPsplice | SOAPsnp

SOAPdenovo

- Introduction »
- System requirements »
- Download »
- Installation »
- Command Line Options »
- FAQ »

Feedback:
soap@genomics.org.cn

Introduction

SOAPdenovo is a novel short-read assembly method that can build a de novo draft assembly for the human-sized genomes. The program is specially designed to assemble Illumina GA short reads. It creates new opportunities for building reference sequences and carrying out accurate analyses of unexplored genomes in a cost effective way. Now the new version is available. SOAPdenovo2, which has the advantage of a new algorithm design that reduces memory consumption in graph construction, resolves more repeat regions in contig assembly, increases coverage and length in scaffold construction, improves gap closing, and optimizes for large genome.

System requirements

SOAPdenovo aims for large plant and animal genomes, although it also works well on bacteria and fungi genomes. It runs on 64-bit Linux system with a minimum of 5G physical memory. For big genomes like human, about 150 GB memory would be required.

Download

The executable program and source code are now available on SourceForge:

Download http://sourceforge.net/projects/soapdenovo2/files/SOAPdenovo2/

Other relative programs(Know more):
SOAPec_v2
Downloa
ErrorCorrec
Download http://sourceforge.net/projects/soapdenovo2/files/ErrorCorrection/ErrorCorrection.tgz/download
GapCloser-bin-v1.12-r6.tgz, a tool named GapCloser for SOAPdenovo:
Download http://sourceforge.net/projects/soapdenovo2/files/GapCloser/
Data prepare module using assembled contig to do scaffold assembly:
Download http://soap.genomics.org.cn/down/prepare.tgz

http://soap.genomics.org.cn/soapdenovo.html

SOAPdenovo

Get it started

Once the configuration file (config_file) is available, a typical way to run the assembler is:

- ☑ ${bin} all -s config_file -K 63 -R -o graph_prefix 1>ass.log 2>ass.err
- ☑ User can also choose to run the assembly process step by step as:
- ☑ step1: ${bin} pregraph -s config_file -K 63 -R -o graph_prefix 1>pregraph.log 2>pregraph.err
- ☑ OR ${bin} sparse_pregraph -s config_file -K 63 -z 5000000000 -R -o graph_prefix 1>pregraph.log 2>pregraph.err
- ☑ step2: ${bin} contig -g graph_prefix -R 1>contig.log 2>contig.err
- ☑ step3: ${bin} map -s config_file -g graph_prefix 1>map.log 2>map.err
- ☑ step4: ${bin} scaff -g graph_prefix -F 1>scaff.log 2>scaff.err

SOAPdenovo

Configuration file

1) avg_ins
This value indicates the average insert size of this library or the peak value position in the insert size distribution figure.

2) reverse_seq
This option takes value 0 or 1. It tells the assembler if the read sequences need to be complementarily reversed.

3) asm_flags
This indicator decides in which part(s) the reads are used. It takes value 1(only contig assembly), 2 (only scaffold assembly), 3(both contig and scaffold assembly), or 4 (only gap closure).

4) rd_len_cutof
The assembler will cut the reads from the current library to this length.

5) rank
It takes integer values and decides in which order the reads are used for scaffold assembly. Libraries with the same "rank" are used at the same time during scaffold assembly.

6) pair_num_cutoff
This parameter is the cutoff value of pair number for a reliable connection between two contigs or pre-scaffolds. The minimum number for paired-end reads and mate-pair reads is 3 and 5 respectively.

7) map_len
This takes effect in the "map" step and is the minimun alignment length between a read and a contig required for a reliable read location.

SOAPdenovo

```
#maximal read length
max_rd_len=100
[LIB]
#average insert size
avg_ins=200
#if sequence needs to be reversed
reverse_seq=0
#in which part(s) the reads are used
asm_flags=3
#use only first 100 bps of each read
rd_len_cutoff=100
#in which order the reads are used while scaffolding
rank=1
# cutoff of pair number for a reliable connection (at least 3 for short insert size)
pair_num_cutoff=3
#minimum aligned length to contigs for a reliable read location (at least 32 for short
insert size)
map_len=32
#a pair of fastq file, read 1 file should always be followed by read 2 file
q1=/path/**LIBNAMEA**/fastq1_read_1.fq
q2=/path/**LIBNAMEA**/fastq1_read_2.fq
```

SOAPdenovo

#another pair of fastq file, read 1 file should always be followed by read 2 file
q1=/path/**LIBNAMEA**/fastq2_read_1.fq
q2=/path/**LIBNAMEA**/fastq2_read_2.fq
#a pair of fasta file, read 1 file should always be followed by read 2 file
f1=/path/**LIBNAMEA**/fasta1_read_1.fa
f2=/path/**LIBNAMEA**/fasta1_read_2.fa
#another pair of fasta file, read 1 file should always be followed by read 2 file
f1=/path/**LIBNAMEA**/fasta2_read_1.fa
f2=/path/**LIBNAMEA**/fasta2_read_2.fa
#fastq file for single reads
q=/path/**LIBNAMEA**/fastq1_read_single.fq
#another fastq file for single reads
q=/path/**LIBNAMEA**/fastq2_read_single.fq
#fasta file for single reads
f=/path/**LIBNAMEA**/fasta1_read_single.fa
#another fasta file for single reads
f=/path/**LIBNAMEA**/fasta2_read_single.fa
#a single fasta file for paired reads
p=/path/**LIBNAMEA**/pairs1_in_one_file.fa
#another single fasta file for paired reads
p=/path/**LIBNAMEA**/pairs2_in_one_file.fa

SOAPdenovo

#bam file for single or paired reads, reads 1 in paired reads file should always be followed by reads 2
NOTE: If a read in bam file fails platform/vendor quality checks(the flag field 0x0200 is set), itself and it's paired
read would be ignored.
b=/path/**LIBNAMEA**/reads1_in_file.bam
#another bam file for single or paired reads
b=/path/**LIBNAMEA**/reads2_in_file.bam
[LIB]
avg_ins=2000
reverse_seq=1
asm_flags=2
rank=2
# cutoff of pair number for a reliable connection (at least 5 for large insert size)
pair_num_cutoff=5
#minimum aligned length to contigs for a reliable read location (at least 35 for large insert size)
map_len=35
q1=/path/**LIBNAMEB**/fastq_read_1.fq
q2=/path/**LIBNAMEB**/fastq_read_2.fq
f1=/path/**LIBNAMEA**/fasta_read_1.fa
f2=/path/**LIBNAMEA**/fasta_read_2.fa

SOAPdenovo

${bin} all -s config_file -K 63 -R -o graph_prefix 1>ass.log 2>ass.err

-s <string> configFile: the config file of solexa reads

-o <string> outputGraph: prefix of output graph file name

-K <int> kmer(min 13, max 63/127): kmer size, [23]

-p <int> n_cpu: number of cpu for use, [8]

-a <int> initMemoryAssumption: memory assumption initialized to avoid further reallocation, unit G, [0]