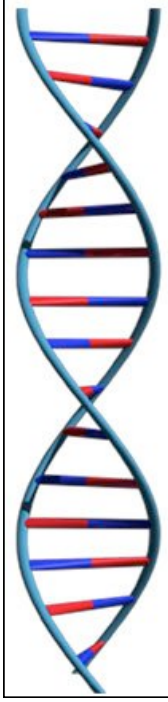
Genome Assembly Tutorial

Deb Triant
Based on the tutorial by Mike Schatz

18 October, 2015
Programming for Biology
Cold Spring Harbor Labs

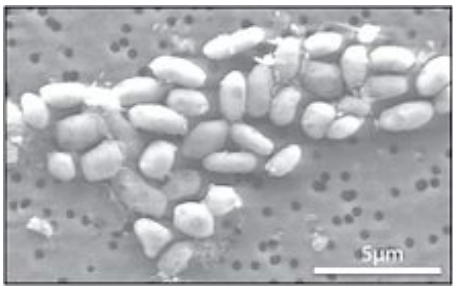


Outline

1. Genome assembly with ALLPATHS-LG
2. Genome alignment with MUMmer

2

Sample Data: Halomonas sp. GFAJ-1



Library 1: Fragment
Avg Read length: 100bp
Insert length: 180bp

Library 2: Short jump
Avg Read length: 50bp
Insert length: 2000bp

A Bacterium That Can Grow by Using Arsenic Instead of Phosphorus
Wolfe-Simon et al (2010) *Science. 332(6034)1163-1166.*

3

Files needed for exercises:

- **File location: /home/pfb2015/data/GenomeAssembly_data/asm**
- **FASTQ files:**
  Paired-end: Illumina 2x100 reads from 180 +/- 20 bp fragments
  1. frag180.1.fq
  2. frag180.2.fq
  Mate-pair/jumping: Illumina 2x50 reads from 2000 +/- 200 bp fragments
  1. jump2k.1.fq
  2. jump2k.2.fq
- **Reference file**
  ref.fa
- **ALLPATHS input files**
  in_groups.csv - name and location of files*
  in_libs.csv - library details*
  *names in both files must match

4

Logging in for graphical display

- ssh -i Student-PFB2015.pem –X deb@compute.programmingforbiology.org
- Exit
- login again with:
  ssh -i Student-PFB2015.pem –AX
  deb@compute.programmingforbiology.org
- Type "xeyes" at terminal to test. Crl-C to end xeyes

5

Sample fastq file

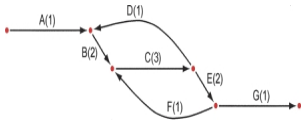
Paired-end

```
@1/1
GGCATGCTGTCATCAAGCGAGTTACCGCTCTACCGCAGCAAACGATGAACCGTCATTACAAACGAACGTTTTA
AACAGGTATTTTTTATCGTACTGTAAA
+
??>>>====<<<<<;;;;;;;;;;;;;;;9999999999888888888888877777777777777777666666666666666666666555555555555
@2/1
ACGCAGCGCGTGTTCCGATGATGACCCAATCCGCAGGCGGCATGTTTGGCGTTTTCTTTACGTCGCAATCCC
GTATGGATAACTTTGCTCAGGCCACCGC
+
??>>>====<<<<<;;;;;;;;;;;;;;;9999999999888888888888877777777777777777666666666666666666666555555555555
```

Mate-pair/Jumping

```
@1/1
CGTACGCCATCATGATGCCCATACCCAGCGACAGCGTGAAGAAAGCCTGG
+
555555666666666677777777788888899999:::::;;;;<<===>>?
@2/1
CCACGGGTCAGGAAAGAACAGCTGCAGCGTAGTCAGCGAGCCCTCTGGCA
+
555555666666666677777777788888899999:::::;;;;<<===>>?
```

6



Running ALLPATHS-LG
Iain MacCallum

BROAD INSTITUTE

7

How to use ALLPATHS-LG

1. **Data requirements (*** most critical ***)**
2. Computational requirements & Installation
3. Preparing your data
4. Assembling
5. What is an ALLPATHS-LG assembly?

8

ALLPATHS-LG sequencing model

Libraries (insert types)	Fragment size (bp)	Read length (bases)	Sequence coverage (x)	Required
Fragment	180*	≥ 100	45	yes
Short jump	3,000	≥ 100 preferable	45	yes
Long jump	6,000	≥ 100 preferable	5	no**
Fosmid jump	40,000	≥ 26	1	no**

*See next slide.

**For best results. Normally not used for small genomes. However essential to assemble long repeats or duplications.

Cutting coverage in half still works, with some reduction in quality of results.

All: protocols are either available, or in progress.

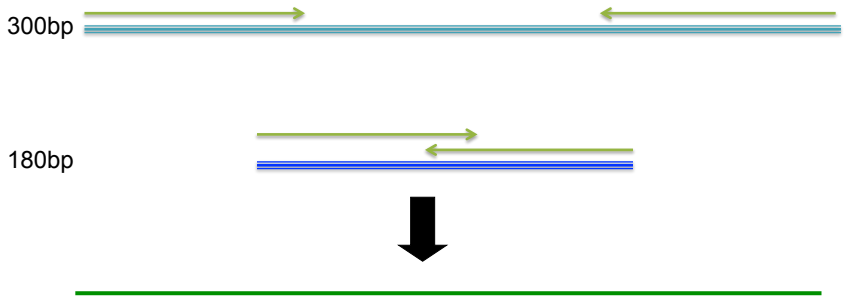
9

Paired-end libraries from 180 bp fragments

Requires short insert separation – less than twice the read length so that reads may overlap.

ALLPATHS tries to close gap in overlapping reads to generate longer reads that are twice as long.

* 300bp with inward orientation – 100bp reads

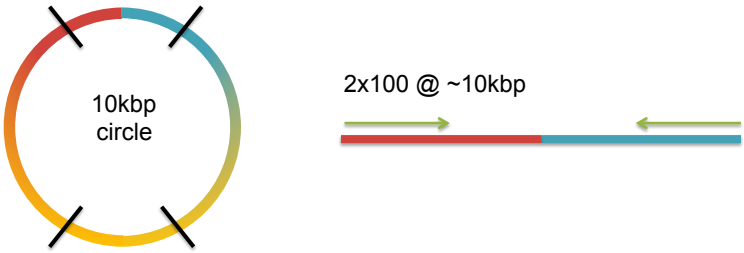


For longer reads, fragment size would be increased proportionally.

10

Mate-pair sequencing

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



11

How to use ALLPATHS-LG

1. Data requirements
2. **Computational requirements & installation**
3. Preparing your data
4. Assembling
5. What is an ALLPATHS-LG assembly?

12

Computational requirements

- **64-bit Linux**
- **runs multi-threaded on a single machine**
- **memory requirements**
  - about 160 bytes per genome base, implying
    - need 512 GB for mammal (Dell R315, 48 processors, $39,000)
    - need 1 GB for bacterium (theoretically)
  - if coverage different than recommended, adjust...
  - potential for reducing usage
- **wall clock time to complete run**
  - 5 Mb genome → 1 hour (8 processors)
  - 2500 Mb genome → 500 hours (48 processors)

13

Installing ALLPATHS-LG

Web page:

http://www.broadinstitute.org/software/allpaths-lg/blog/

General instructions:

http://www.broadinstitute.org/science/programs/genome-biology/computational-rd/general-instructions-building-our-software

USE THE MANUAL!!!

14

Getting the ALLPATHS-LG source

****Do not need to do for course exercise.....just for reference****

Our current system is to release code daily if it passes a test consisting of several small assemblies:

Download the latest build from:
ftp://ftp.broadinstitute.org/pub/crd/ALLPATHS/Release-LG/

Unpack it:

```
% tar xzf allpathslg-39099.tar.gz
```

(substitute the latest revision id for 39099)

This creates a source code directory `allpathslg-39099`:

```
% cd allpathslg-39099
```

15

Building ALLPATHS-LG

Step one: `./configure`

Options:
`-prefix=<prefix path>`
put binaries in `<prefix path>/bin`, else `./bin`

Step two: `make` and `make install`

Options:
`-j<n>`
compile with n parallel threads

Step three: add bin directory to your path

16

How to use ALLPATHS-LG

1. Data requirements
2. Computational requirements & Installation
3. **Preparing your data** - `PrepareAllPathsInputs.pl`
4. Assembling
5. What is an ALLPATHS-LG assembly?

17

Preparing data for ALLPATHS-LG

Before assembling, prepare and import your read data.

ALLPATHS-LG expects reads from:

- At least one fragment library.
  - One should come from fragments of size ~180 bp.
  - This is NOT checked but otherwise results will be bad.
- At least one long-insert library.

IMPORTANT: use all the reads, including those that fail the Illumina purity filter. These low quality reads may cover 'difficult' parts of the genome.

Trimming reads?

18

ALLPATHS-LG input format

ALLPATHS-LG can import data from:
BAM, FASTQ, FASTA/QUALA or FASTB/QUALB files.

You must also provide two metadata files to describe them:

`in_groups.csv` - ties files to libraries
`in_libs.csv` - describes the libraries

FASTQ format: consists of records of the form
@<read name>
<sequence of bases, multiple lines allowed>
+
<sequence of quality scores, with Qn represented by ASCII code n+33, multiple lines allowed>

19

Sample fastq file

Paired-end

```
@1/1
GGCATGCTGTCATCAAGCGAGTTACCGCTCTACCGCAGCAAACGATGAACCGTCATTACAAACGAACGTTTTA
AACAGGTATTTTTTATCGTACTGTAAA
+
??>>>====<<<<<;;;;;;;;;;;;;;;;999999999988888888888887777777777777777666666666666666666666555555555555
@2/1
ACGCAGCGCGTGTTCCGATGATGACCCAATCCGCAGGCGGCATGTTTGGCGTTTTCTTTACGTCGCAATCCC
GTATGGATAACTTTGCTCAGGCCACCGC
+
??>>>====<<<<<;;;;;;;;;;;;;;;;999999999988888888888887777777777777777666666666666666666666555555555555
```

Mate-pair/Jumping

```
@1/1
CGTACGCCATCATGATGCCCATACCCAGCGACAGCGTGAAGAAAGCCTGG
+
555555666666666677777777788888899999:::::;;;;<<===>>?
@2/1
CCACGGGTCAGGAAAGAACAGCTGCAGCGTAGTCAGCGAGCCCTCTGGCA
+
555555666666666677777777788888899999:::::;;;;<<===>>?
```

20

Input files – in_groups.csv

Each line in **in_groups.csv** comma separated value file, corresponds to a BAM or FASTQ file you wish to import for assembly.
The library name must match the names in **in_libs.csv**.

group_name - a unique nickname for this file
library_name - library to which the file belongs
file_name - the absolute path to the file
(should end in .bam or .fastq)
(use wildcards '?', '*' for paired fastqs)

Example:

```
group_name, library_name,                    file_name
302GJ,      Solexa-11541, /seq/Solexa-11541/302GJABXX.bam
303GJ,      Solexa-11623, /seq/Solexa-11623/303GJABXX.?.fastq
```

21

Input files – in_groups.csv

Each line in **in_groups.csv** comma separated value file, corresponds to a BAM or FASTQ file you wish to import for assembly.
The library name must match the names in **in_libs.csv**.

group_name - a unique nickname for this file
library_name - library to which the file belongs
file_name - the absolute path to the file
(should end in .bam or .fastq)
(use wildcards '?', '*' for paired fastqs)

Example:

```
group_name, library_name,                    file_name
302GJ,      Solexa-11541, /seq/Solexa-11541/302GJABXX.bam
303GJ,      Solexa-11623, /seq/Solexa-11623/303GJABXX.?.fastq
```

22

Libraries - in_libs.csv

EXAMPLE

```
library_name,     type, paired, frag_size, frag_stddev, insert_size, insert_stddev, read_orientation, genomic_start, genomic_end
Solexa-11541, fragment,      1,       180,          10,           ,            ,            inward              ,
Solexa-11623,  jumping,      1,         ,           ,        3000,          500,           outward              0,          25
```

Pay attention to commas and empty fields

23

Libraries – in_libs.csv

For fragment libraries only

`frag_size` - estimated mean fragment size

`frag_stddev` - estimated fragment size std dev

For jumping libraries only

`insert_size` - estimated jumping mean insert size

`insert_stddev` - estimated jumping insert size std dev

These values determine how a library is used. If `insert_size` is ≥ 20,000, the library is assumed to be a Fosmid jumping library.

`paired` - always 1 (only supports paired reads)

`read_orientation` - `inward` or `outward`.

Paired reads can either point towards each other, or away from each other. Currently fragment reads must be `inward`, jumping reads `outward`, and Fosmid jumping reads `inward`.

24

Libraries – in_libs.csv

Reads can be trimmed to remove non-genomic bases produced by the library construction method:

genomic_start - inclusive zero-based range of read bases
genomic_end to be kept; if blank or 0 keep all bases

Reads are trimmed in their original orientation.

Extra optional fields (descriptive only – ignored by ALLPATHS)

project_name - a string naming the project.
organism_name - the organism name.
type - fragment, jumping, EcoP15I, *etc*.

EXAMPLE

```
library_name,     type, paired, frag_size, frag_stddev, insert_size, insert_stddev, read_orientation, genomic_start, genomic_end
Solexa-11541, fragment,      1,       180,          10,           ,             ,           inward              ,          24
Solexa-11623,  jumping,      1,          ,           ,        3000,           500,          outward             0,          25
```

How to import assembly data files

PrepareAllPathsInputs.pl

```
IN_GROUPS_CSV=<in groups file>
IN_LIBS_CSV=<in libs file>
DATA_DIR=<full path of data directory>  *Required
PLOIDY=<ploidy, either 1 or 2>
PICARD_TOOLS_DIR=<picard tools directory> *in your
path - do not need to edit
```

- **IN_GROUPS_CSV** and **IN_LIBS_CSV**: **/assolute_path/in_groups.csv** and **/absolute_path/in_libs.csv.** These arguments determine where the data are found.

- **DATA_DIR**: imported data will be placed here. Want to name your DATA_DIR with the same pathway as will use in RunAllPathsLG for assembly.

- **PLOIDY**: either 1 (for a haploid or inbred organism), or 2 (for a diploid organism) .

- **PICARD_TOOLS_DIR**: path to Picard tools, for data conversion from BAM. Do not need to specify if in path.

26

Putting it all together

1. Create your input files:
   in_libs.csv - data file to describe your libraries and a
   in_groups.csv - data file to describe your data files.

2. Prepare input files

```
% cd /path/to/data/
% PrepareAllPathsInputs.pl \
    DATA_DIR=data_dir PLOIDY=1 >& prepare.log
```

Want to name your DATA_DIR with the same pathway as will use in RunAllPathsLG for assembly: <PRE>/<REFERENCE>/<DATA>/<RUN>/ASSEMBLIES/test

```
CSHL/Halomonas/mydata/     #where to save your data

% mkdir -p CSHL/Halomonas/mydata  # -p for making parent dirs
Our data:
% PrepareAllPathsInputs.pl \
      DATA_DIR=abs_path/CSHL/Halomonas/mydata/ PLOIDY=1 >&
   prepare.log #make sure you run in dir containing input files
```

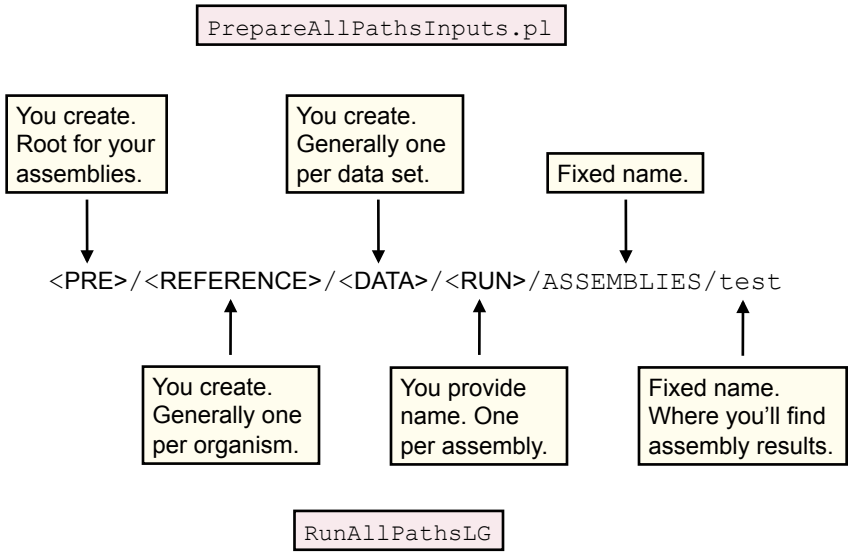
27

How to use ALLPATHS-LG

1. Data requirements
2. Computational requirements & installation
3. Preparing your data
4. **Assembling** - RunAllPathsLG
5. What is an ALLPATHS-LG assembly?

28

ALLPATHS-LG directory structure



29

How to assemble

To run asssembly

```
% RunAllPathsLG                  \
   PRE=<prefix path>             \
   REFERENCE_NAME=<reference dir>    \
   DATA_SUBDIR=<data dir>          \
   RUN=<run dir>
```

Automatic resumption. If the pipeline crashes, fix the problem, then run the same `RunAllPathsLG` command again. Execution will resume where it left off. `% OVERWRITE=True`

What can go wrong: Not enough RAM, CPU time, artifacts or contamination in your data. Log files very informative!

Results. The assembly files are:

`final.contigs.fasta`	- fasta contigs
`final.contigs.efasta`	- efasta contigs
`final.assembly.fasta`	- scaffolded fasta
`final.assembly.efasta`	- scaffolded efasta

30

Putting it all together

1. Collect the BAM or FASTQ files that you wish to assemble. Create a
   in_libs.csv metadata file to describe your libraries and a
   in_groups.csv metadata file to describe your data files.

2. Prepare input files

```
% PrepareAllPathsInputs.pl \
  DATA_DIR=abs_path/CSHL/Halomonas/mydata/ PLOIDY=1 >&
  prepare.log  **This is where PrepareAllPathsInputs.pl
  saves our input files: CSHL/Halomonas/mydata/
```

3. Assemble.

```
% RunAllPathsLG \
   PRE= REFERENCE_NAME= \
   DATA_SUBDIR= RUN= THREADS=4 >& run.log
```

Our assembly example:

```
RunAllPathsLG \
   PRE=CSHL REFERENCE_NAME=Halomonas \
   DATA_SUBDIR=mydata RUN=run-1 THREADS=1 >& run-1.log
```

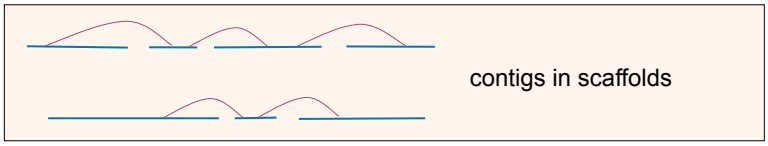
31

How to use ALLPATHS-LG

1. Data requirements
2. Computational requirements & installation
3. Preparing your data
4. Assembling
5. **What is an ALLPATHS-LG assembly?**

32

1. Linear assemblies



contig: a contiguous sequence of bases….

```
CTGCCCCCTGTGCCAATGGGTTTGAGGCTCTTCCCACTTCCTTTTCTATTAGATTCAATGTATCTGGTTTTATGTTGAGG
TCCTAGATCCACTTGGACTTGAGCTTTGTACAAGATGACATATATAGGTCTGTTTTTATTCTTCTACATACAGACAGCCA
GTTATACCAGCACCATTTATTGAAGACACTTTCTTTATTCCATTGTATATTTTTTTACTTCCTTGTCAAAAATCAAGTGA
CCATGAGTATGTGGTTTCATTTCTGGGTCTTCAATTGTATTCCATTAGTCAACATATCTGTCTCTGTACCAATACCATGC
```

scaffold: a sequence of contigs, separated by gaps….

```
TCCTAGATCCACTTGGACTTGAGCTTTGTACAAGATGACATATATAGGTCTGTTTTTATTCTTCTACATACAGACAGCCA
GTTATACCAGCACCATTTATTGAAGACACTTTCTTTATTCCATTGTATATTTTTTTACTTCCTTGTCAAAAATCAAGTGA
CCATGAGTATGTGGTTTCATTTCTGGGTCTTCAATTGTATTCCATTAGTCAACATATCTGTCTCTGTACCAATACCATGC
NNNNNNNN
AGTTTTTACCACAATTGCTCTATAGTAAAGCTTGAGGTCAGGGTTGGTGATCCCTCCAGCCATTCTTTCATTATTAAGAA
TTGTTTTCCCTAGTCTGGGTTTTTTGCTTTTCCAGGCGAATTTGAGAATTGCTCTTTCCATGTCTTTGAAGAATTGTGTT
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
GGGATTTTGATGGGGTTTGCATTGAATCTGTAGATTGTCTTTGGTAAGATGGTTAGTTTTACTATGTTAATTCTGCCAAT
CCACAAGCATGGGAGCGCTCTCCATTTTCTGAGATCTTCTTCAATTTCTTTCTTGAGAAACTTGAAGTTATTGTCATACA
```

Number of Ns = predicted gap size, with error bars (can't be displayed in fasta format)

33

1. Linear assemblies

Example of an assembly in fasta format

```
>scaffold_1
TCCTAGATCCACTTGGACTTGAGCTTTGTACAAGATGACATATATAGGTCTGTTTTTATTCTTCTACATACAGACAGCCA
GTTATACCAGCACCATTTATTGAAGACACTTTCTTTATTCCATTGTATATTTTTTTACTTCCTTGTCAAAAATCAAGTGA
CCATGAGTATGTGGTTTCATTTCTGGGTCTTCAATTGTATTCCATTAGTCAACATATCTGTCTCTGTACCAATACCATGC
NNNNNNNN
AGTTTTTACCACAATTGCTCTATAGTAAAGCTTGAGGTCAGGGTTGGTGATCCCTCCAGCCATTCTTTCATTATTAAGAA
TTGTTTTCCCTAGTCTGGGTTTTTTGCTTTTCCAGGCGAATTTGAGAATTGCTCTTTCCATGTCTTTGAAGAATTGTGTT
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
GGGATTTTGATGGGGTTTGCATTGAATCTGTAGATTGTCTTTGGTAAGATGGTTAGTTTTACTATGTTAATTCTGCCAAT
CCACAAGCATGGGAGCGCTCTCCATTTTCTGAGATCTTCTTCAATTTCTTTCTTGAGAAACTTGAAGTTATTGTCATACA
>scaffold_2
CTGAAGTTGTTTATCAGCTGGAGAAGTTCTCAGGTAGAATTTTTGGGATTGCTTATGTATGCTATCATATCACTTGCAAA
TAGTGATACCTTGATTTCTTTTTTACCAATATGTATCCCATTGATCTCTTTCTGTTGTCTTATTGTTCTAGCTAACACTT
CAAGTACTATATTGAATAGATATGGGGAGAGTGGGAATCCTTGTCTTGTCTCCGATTTCAGTGGGATTGCTTCAAGTATG
```

34

3. Linearized graph assemblies

Efasta: extended fasta

...ACTGTTT{A,C}GAAAT... A or C at site

...CGCGTTTTTTTTTT{,T,TT}CAT... 0 or 1 or 2 Ts at site

Example of an assembly in efasta format

```
>scaffold_1
TCCTAGATCCACTTGGACTTGAGCTTTGTATATATATATATATATATATA{,TA}CAAGATGACATATATAGGAGACAGCCA
GTTATACCAGCACCATTTATTGAAGACACTTTCTTTATTCCATTGTATATTTTTTTACTTCCTTGTCAAAAATCAAGTGA
CCATGAGTATGTGGTTTCATTTCTGGGTCTTCAATTGTATTCCATTAGTCAACATATCTGTCTCTGTACCAATACCATGC
NNNNNNNN
AGTTTTTACCACAATTGCTCTATAGTAAAGCTTGAGGTCAGGGTTGGTGATCCCTCCAGCCATTCTTTCATTATTAAGAA
TTGTTTTCCCTAGTCTGGGTTTTTTGCTTTTCCAGGCGAATTTGAGAATTGCTCTTTCCATGTCTTTGAAGAATTGTGTT
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
GGGATTTTGATGGGGTTTGCATTGAATCTGTAGATTGTCTTTGGTAAGATGGTTAGTTTTACTATGTTAATTCTGCCAAT
CCACAAGCATGGGAGCGCTCTCCATTTTCTGAGATCTTCTTCAATTTCTTTCTTGAGAAACTTGAAGTTATTGTCATACA
>scaffold_2
CTGAAGTTGTTTATCAGCTGGAGAAGTTCTCAGGTAGAATTTTTGGGATT{A,C,G}GCTTATGTATGCTATCTTGCAAA
TAGTGATACCTTGATTTCTTTTTTACCAATATGTATCCCATTGATCTCTTTCTGTTGTCTTATTGTTCTAGCTAACACTT
CAAGTACTATATTGAATAGATATGGGGAGAGTGGGAATCCTTGTCTTGTCTCCGATTTCAGTGGGATTGCTTCAAGTATG
```

35

Putting it all together

1. Collect the BAM or FASTQ files that you wish to assemble. Create a **in_libs.csv** metadata file to describe your libraries and a **in_groups.csv** metadata file to describe your data files.

2. Prepare input files

```
% PrepareAllPathsInputs.pl \
    DATA_DIR=`pwd` PLOIDY=1 >& prepare.log
```

3. Assemble.

```
% RunAllPathsLG \
    PRE=. REFERENCE_NAME=. \
    DATA_SUBDIR=. RUN=default THREADS=4 >& run.log
```

4. Get the results (four files).

```
% cd default/ASSEMBLIES/test/
% less final.{assembly,contigs}.{fasta,efasta}
```

Once input files are generated (PrepareAllPathsInputs.pl) can run assemblies with different parameters.

36

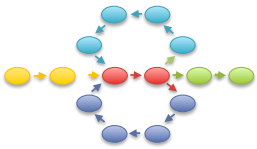
Assembly parameters

- Parameters – all listed in ALLPATHS manual
- What about kmers?

"The user should **not** adjust the kmer size from the default value of K=96.

The relationship between kmer size K and read size is not a direct one in ALLPATHS-LG, unlike in may other assemblers. ALLPATHS-LG actually uses a number of different sizes of K internally, and because of this, it is not intended that users change the K values for an assembly".

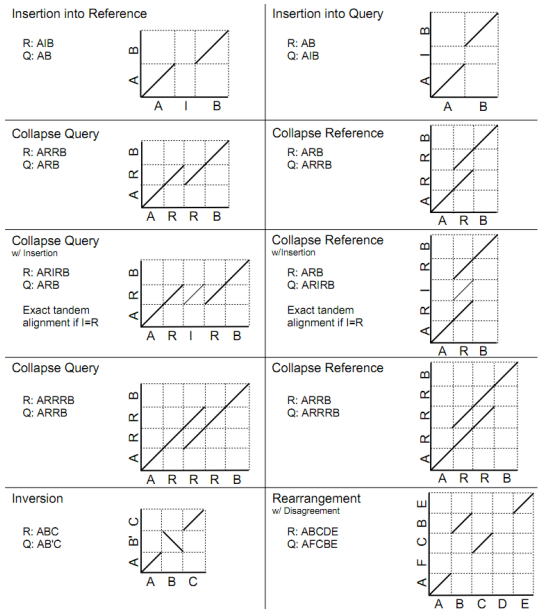
37



Whole Genome Alignment with MUMmer

38

Structural Variant Types



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

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

39

Using MUMmer

- **mummer** – core program of package. Outputs matches between reference and query
- **nucmer** - all-vs-all comparison of nucleotide sequences contained in fasta files. Best for highly similar sequences with possible rearrangements
- **mummerplot** – generates dotplots and coverage plots
- **show-coords** – parses the output of nucmer and displays the coordinates

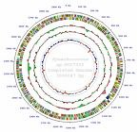
40

Using MUMmer

```
mummer ref.fa CSHL/Halomonas/mydata/run-1/ASSEMBLIES/test/ \
   final.contigs.fasta > mummer.out

nucmer --maxmatch ref.fa \
CSHL/Halomonas/mydata/run-1/ASSEMBLIES/test/
   final.contigs.fasta -p refctg
   -maxmatch     Find maximal exact matches (MEMs) without repeat filtering
   -p refctg     Set the output prefix for delta file

mummerplot --layout refctg.delta
   -r   Show the dotplot

show-coords -rclo refctg.delta
   -r   Sort alignments by reference position
   -c   Show percent coverage
   -l   Show sequence lengths
   -o   Annotate each alignment with BEGIN/END/CONTAINS
```

41

Resources



- Assembly Competitions
  - Assemblathon: http://assemblathon.org/
  - GAGE: http://gage.cbcb.umd.edu/
- Assembler Websites:
  - ALLPATHS-LG: http://www.broadinstitute.org/software/allpaths-lg/blog/
  - SOAPdenovo: http://soap.genomics.org.cn/soapdenovo.html
  - Celera Assembler: http://wgs-assembler.sf.net
- Tools:
  - MUMmer: http://mummer.sourceforge.net/
  - Quake: http://www.cbcb.umd.edu/software/quake/
  - AMOS: http://amos.sf.net

42