Long Interval Nucleotide K-mer Scaffolder
LINKS v1.7

René L. Warren, 2014-2016
email: rwarren at bcgsc.ca

Name

LINKS: Long Interval Nucleotide K-mer Scaffolder

Description

LINKS is a genomics application for scaffolding genome assemblies with long reads, such as those produced by Oxford Nanopore Technologies Ltd. It can be used to scaffold high-quality draft genome assemblies with any long sequences (eg. ONT reads, PacBio reads, another draft genomes, MPET etc)

What's new in v1.7 ?

Support for scaffolding with MPET (jumping library) reads

Support for reading compressed long sequence [reads] and assembly files

Implemented mid-scaffolding checkpoint to more quickly test certain parameters (-l min. links / -a min. links ratio), recover from crash and explore very large kmer space

What's new in v1.6 ?

Incorporation of the BC Genome Sciences Centre custom Bloom filter with the rolling hash function. This new data structure supports the creation of Bloom filters from large genome assemblies (tested on assemblies of 3 Gbp human and 20 Gbp white spruce). The Bloom filter data structure swap in v1.6 offers a ~30-fold kmer insert speed-up (~6x query speed-up) over v1.5.2, while supporting the creation of filters from large genome assembly drafts

What's new in v1.5.2 ?

LINKS outputs a scaffold graph in gv format, with highlighted merges and edge attributes

What's new in v1.5.1 ?

Fixed a bug that prevented the creation of Bloom filters with a different false positive rate (FPR) than default. Using lower FPR does not influence scaffolding itself, only run time. For large genomes (>1Gbp), using a higher FPR is recommended when compute memory (RAM) is limiting.

What's new in v1.5 ?

LINKS uses a Bloom filter to limit hashed paired k-mers to only those found in the sequence file to re-scaffold. This feature decreases RAM usage by over 60%, while the run time is nearly unchanged. When ran iteratively, users can re-use Bloom filters with the -r options, which results in faster run times up to half compared to v1.3 and earlier.

What's new in v1.3 ?

Added support for fastq files. Added support for multiple long-reads files. With v1.3, the reads file is not supplied directly through -s, but with a file-of-filenameinstead, which is a text file listing the fullpath/FASTA or FASTQ on your system. The file-of-filenames supplied through the -s option could include a mixture of FASTA and FASTQ files.

What's new in v1.2 ?

Fixed bug that prevented reading traditional FASTA sequences (where a sequence is represented as a series of linestypically no longer than 120 characters)

What's new in v1.1 ?

Included offset option (-o option) - Enable LINKS to explore a wider k-mer space range when running iteratively Minor fixes: IUPAC codes are now preserved

Implementation and requirements

LINKS is implemented in PERL and runs on any OS where PERL is installed. In v1.6, there is a single dependency to the BloomFilter.pm (included, see below for download/compile instructions) - BC Genome Sciences Centre's common Bloom filter In v1.5, there is a single dependency to Bloom::Faster - an extension for the c library libbloom.

Install

Download the tar ball, gunzip and extract the files on your system using:
gunzip links_v1-7.tar.gz tar -xvf links_v1-7.tar
In v1.6+, the use of the Bloom::Faster PERL library is deprecated

INSTRUCTIONS TO BUILD THE BloomFilter PERL module

1. DOWNLOAD the BC Genome Sciences Centre's BloomFilter: The BTL C/C++ Common Bloom filters for bioinformatics projects, as well as any APIs created for other programming languages.
   cd ./links_v1.7/lib
   git clone git://github.com/bcgsc/bloomfilter.git
   cd swig
2. BUILD a PERL5 module

Make sure you have swig installed and included in your path.

http://www.swig.org/

TO BUILD a Perl5 module (run in swig/):

```
a) preinst-swig -Wall -c++ -perl5 BloomFilter.i
b) g++ -c BloomFilter_wrap.cxx -I/usr/lib64/perl5/CORE -fPIC –O3
c) Dbool=char g++ -Wall -shared BloomFilter_wrap.o -o BloomFilter.so –
O3
```

TO COMPILE, swig needs the following Perl5 headers:

```
#include "Extern.h" #include "perl.h" #include "XSUB.h"
```

If they are not located in /usr/lib64/perl5, you can run "perl -e 'use Config; print $Config{archlib};" to locate them.

1. VERIFY your install
   [swig]$ ./test.pl
2. CHANGE the path to BloomFilter.pm in
   LINKS/writeBloom.pl/testBloom.pl

You only need to change if you have re-built in a relative directory different from: use lib "$FindBin::Bin/./lib/bloomfilter/swig"; (for LINKS) use lib "$FindBin::Bin/../lib/bloomfilter/swig"; (for writeBloom.pl/testBloom.pl)

Documentation

Refer to the LINKS-readme.txt and LINKS-readme.pdf file on how to run LINKS and the LINKS web site for information about the software and its performance www.bcgsc.ca/bioinfo/software/links
Questions or comments? We would love to hear from you!
Email: rwarren at bcgsc.ca

Citing LINKS

René L. Warren, Chen Yang, Benjamin P. Vandervalk, Bahar Behsaz, Albert Lagman, Steven J. M. Jones and Inanç Birol. 2015. LINKS: Scalable, alignment-free scaffolding of draft genomes with long reads. GigaScience 4:35
DOI: 10.1186/s13742-015-0076-3 © Warren et al. 2015
Thank you for using, developing and promoting this free software.

Credits

LINKS René Warren
SWIG/BloomFilter.pm Sarah Yeo, Justin Chu
https://github.com/bcgsc/bloomfilter Justin Chu, Ben Vandervalk, Hamid Mohamadi (ntHash), Sarah Yeo, Golnaz Jahesh

Running LINKS

e.g. /usr/bin/time -v -o timeLINKS_ECK12singleTIG.txt ../LINKS -f ecoliK12_abyss_illumina_contig_baseline.fa -s K12_F2D.fof -b ecoliK12-ONT_linksSingleIterationTIG

```
Usage: ./LINKS [v1.7]
-f  sequences to scaffold (Multi-FASTA format, required)
-s  file-of-filenames, full path to long sequence reads or MPET pairs [see
below] (Multi-FASTA/fastq format, required)
-m  MPET read length (default -m 0, optional)
```

```
        ! DO NOT SET IF NOT USING MPET. WHEN SET, LINKS WILL EXPECT A SPECIAL
FORMAT UNDER -s
        ! Paired MPET reads in their original outward orientation <- -> must be
separated by ":"
          >template_name
          CTATGCATAAGCAGACGAGCAGCGACGCAGCACG:ATATATAGCGCACGACGCAGCACAC
-d  distance between k-mer pairs (eg. target distance to re-scaffold on.
default -d 4000, optional)
-k  k-mer value (default -k 15, optional)
-t  step of sliding window when extracting k-mer pairs from long reads (default
-t 2, optional)
-o  offset position for extracting k-mer pairs (default -o 0, optional)
-e  error (%) allowed on -d distance   e.g. -e 0.1  == distance +/- 10%
(default -e 0.1, optional)
-l  minimum number of links (k-mer pairs) to compute scaffold (default -l 5,
optional)
-a  maximum link ratio between two best contig pairs (default -a 0.3, optional)
         *higher values lead to least accurate scaffolding*
-b  base name for your output files (optional)
-r  Bloom filter input file for sequences supplied in -s (optional, if none
provided will output to .bloom)
         NOTE: BLOOM FILTER MUST BE DERIVED FROM THE SAME FILE SUPPLIED IN -f
WITH SAME -k VALUE
         IF YOU DO NOT SUPPLY A BLOOM FILTER, ONE WILL BE CREATED (.bloom)
-p  Bloom filter false positive rate (default -p 0.0001, optional; increase to
prevent memory allocation errors)
-x  Turn off Bloom filter functionality (-x 1 = yes, default = no, optional)
-v  Runs in verbose mode (-v 1 = yes, default = no, optional)

Notes: -s K12_F2D.fof specifies a file-of-filenames (text file) listing:
K12_full2dONT_longread.fa (see ./test folder) -x When turned on (-x 1), LINKS
will run with a behaviour equivalent to v1.3 (no Bloom filters).
This may be useful for large genome assembly drafts and when long reads are
extremely high quality.
```

Tips to minimize memory usage

The most important parameters for decreasing RAM usage are -t and -d. The largest dataset used for scaffolding by our group was a draft assembly of the white spruce genome (20 Gb)* - For this, a large sliding window, -t (200) was used and was decreased as the k-mer distance -d increased. *refer to LINKSrecipe_pglaucaPG29-WS77111.sh in the ./test folder

Because you want want to start with a low -d for scaffolding, you have to estimate how many minimum links (-l) would fit in a -d window +/- error -e given sliding window -t. For instance, it may not make sense to use -t 200, -d 500 at low coverages BUT if you have at least 10-fold coverage it might since, in principle, you should be able to derive sufficient k-mer pairs within same locus if there's no bias in genome sequencing.

For re-scaffolding white spruce, only 1X coverage was available (since the re-scaffolding used a draft assembly instead of long reads), but even -t 200 -d 5000 (1st iteration) did merge scaffolds even though, in theory, the -e parameter will play an important role limiting linkages outside of the target range -d (+/-) -e %. This is especially true when using raw MPET for scaffolding, to limit spurious linkages by contaminating PETs.
On the data side of things, reducing the coverage (using less long reads), and limiting to only the highest quality reads would help decrease RAM usage.
In v1.5, LINKS builds a Bloom filter that comprises all k-mer of a supplied (-f) genome draft and uses it to only hash k-mer pairs from longreads having an equivalent in the Bloom filter. When LINKS runs iteratively, the bloom filter built at the first iteration is re-used thus saving execution time.
These are the best tips I can offer at the moment, until we address it further programmatically using even more efficient data structures & code.

Test data

Go to ./test
To reproduce all the assemblies from the manuscript, execute: ./runall.sh

run:

./runme_EcoliK12single.sh
The script will download the baseline E. coli abyss scaffold assembly and full 2D ONT reads (Quick et al 2014) and used the latter to re-scaffold the former, with default parameters (Table 1D in paper).
NEED ~8GB RAM WITH CURRENT PARAMETERS. Increase (-t) to use less RAM.
./runme_EcoliK12singleMPET.sh will scaffold using E. coli K12 MPET reads (~42 to 90 GB RAM for trimmed vs raw MPET)

./runme_EcoliK12iterative.sh

The script will download the baseline E. coli abyss scaffold assembly and full 2D ONT reads (Quick et al 2014) and used the latter to re-scaffold the former, iteratively 30 times increasing the distance between k-mer pairs at each iteration (Table 1F in paper).

NEED ~16GB RAM WITH CURRENT PARAMETERS. Increase (-t) to use less RAM.

./runme_ScerevisiaeW303iterative.sh

This script will download the S. cerevisiae W303 raw ONT long reads and used them iteratively to scaffold a baseline IlluminaMiSeq assembly (both data from http://schatzlab.cshl.edu/data/nanocorr/). You will need a computer with at least 132GB RAM. This process was clocked at 6:08:21 (h:mm:ss wall clock) and used 118GB RAM on a Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz 16 dualcore (but running on a single CPU). (Fig 1 in the main ms, FigS8 in preprint).

NEED <132GB RAM WITH CURRENT PARAMETERS. Increase (-t) to use less RAM.

./runme_ScerevisiaeS288citerative.sh

This script will download the S. cerevisiae W303 raw ONT long reads and used them iteratively to scaffold a baseline ABySS assembly of Illumina data. You will need a computer with at least 132GB RAM. (Fig 1 in the main ms, FigS8 in preprint).

NEED <132GB RAM WITH CURRENT PARAMETERS. Increase (-t) to use less RAM.

./runme_StyphiH58iterative.sh

This script will download the S. typhi H58 2D ONT long reads and used them iteratively to scaffold a baseline assembly

of Illumina data (both from Ashton,P.M. 2015. Nat.Biotechnol.33,296–300). You will need a computer with at least 132GB RAM. (Fig 1 in the main ms, FigS8 in preprint).

Additional info:
The file: LINKSrecipe_pglaucaPG29-WS77111.sh is provided to show the re-scaffolding recipe used to produce a re-scaffolded white spruce (P. glauca) genome assembly.
Likewise: LINKSrecipe_ecoliRawR7-3.sh is provided to show the process of scaffolding iteratively the E.coli assembly (Table 1H in Warren et al. 2015 manuscript). 30 iterations were done for the paper.

Testing the Bloom filters

Insertions:

cd tools
./writeBloom.pl
Usage: ./writeBloom.pl
-f sequences to scaffold (Multi-FASTA format, required)
-k k-mer value (default -k 15, optional)
-p Bloom filter false positive rate (default -p 0.0001, optional - increase to prevent memory allocation errors)

Queries:

cd tools
./testBloom.pl
Usage: ./testBloom.pl
-f sequences to test (Multi-FASTA format, required)
-k k-mer value (default -k 15, optional)
-r Bloom filter file

How it works

Process: long sequences are supplied as input (-s option, FASTA format) and k-mer pairs are extracted using user-defined k-mer length (-k) and distance between the 5'-end of each pairs (-d) over a sliding window (-t). Unique k-mer pairs at set distance are hashed. FASTA sequences to scaffold are supplied as input (-f), and are shredded to k-mers on both strands, tracking the [contig] sequence of origin, k-mer positions and frequencies of observation.

Algorithm:

FASTA sequences to scaffold are supplied as input (-f), and are shredded to k-mers on both strands, populating a Bloom filter15 whose number of elements corresponds to a rough approximation of the number of k-mers in the draft genome based on file size. The size of the filter can be adjusted by controlling its false positive rate (-p). Building a Bloom filter is optional (-x), but strongly recommended as it decreases the memory usage and run time. ONT reads are supplied as input (-s option, file-of-filenames listing FASTA/FASTQ formatted files) and k-mer pairs are extracted using user-defined k-mer length (-k) and distance between the 5'-end of each pairs (-d) over a sliding window (-t). When both k-mers are found in the Bloom filter, unique k-mer pairs at set distance are hashed, tracking the contig or scaffold of origin, k-mer positions and frequencies of observation. LINKS has two main stages: contig pairing, and scaffold layout. Cycling through k-mer pairs, k-mers that are uniquely placed on contigs are identified. Putative contig pairs are formed if k-mer pairs are on different contigs. Contig pairs are only considered if the calculated distances between them satisfy the mean distance provided (-d), while allowing for a deviation (-e). Contig pairs having a valid gap or overlap are allowed to proceed to the scaffolding stage. Contigs in pairs may be ambiguous: a given contig may link to multiple contigs. To mitigate, the number of spanning k-mer pairs (links) between any given contig pair is

recorded, along with a mean distance estimate. Once pairing between contigs is complete, the scaffolds are built using contigs in turn until all have been incorporated into a scaffold. Scaffolding is controlled by merging sequences only when a minimum number of links (-l) join two contig pairs, and when links are dominant compared to that of another possible pairing (-a). The predecessor of LINKS is the unpublished scaffolding engine in the widely-used SSAKE assembler16, and foundation of the SSPACE-LongRead scaffolder8. A summary of the scaffold layout is provided (.scaffold) as a text file, and captures the linking information of successful scaffolds. A FASTA file (.scaffold.fa) is generated using that information, placing N-pads to represent the estimated lengths of gaps, and a single “n” in cases of overlaps between contigs. A log summary of k-mer pairing in the assembly is provided (.log) along with a text file describing possible issues in pairing (.pairing_issues), pairing distribution (.pairing_distribution.csv) and compressed Bloom filter (.bloom). The Bloom filter is intended to be re-used (supplied via -r) for iterative LINKS runs.

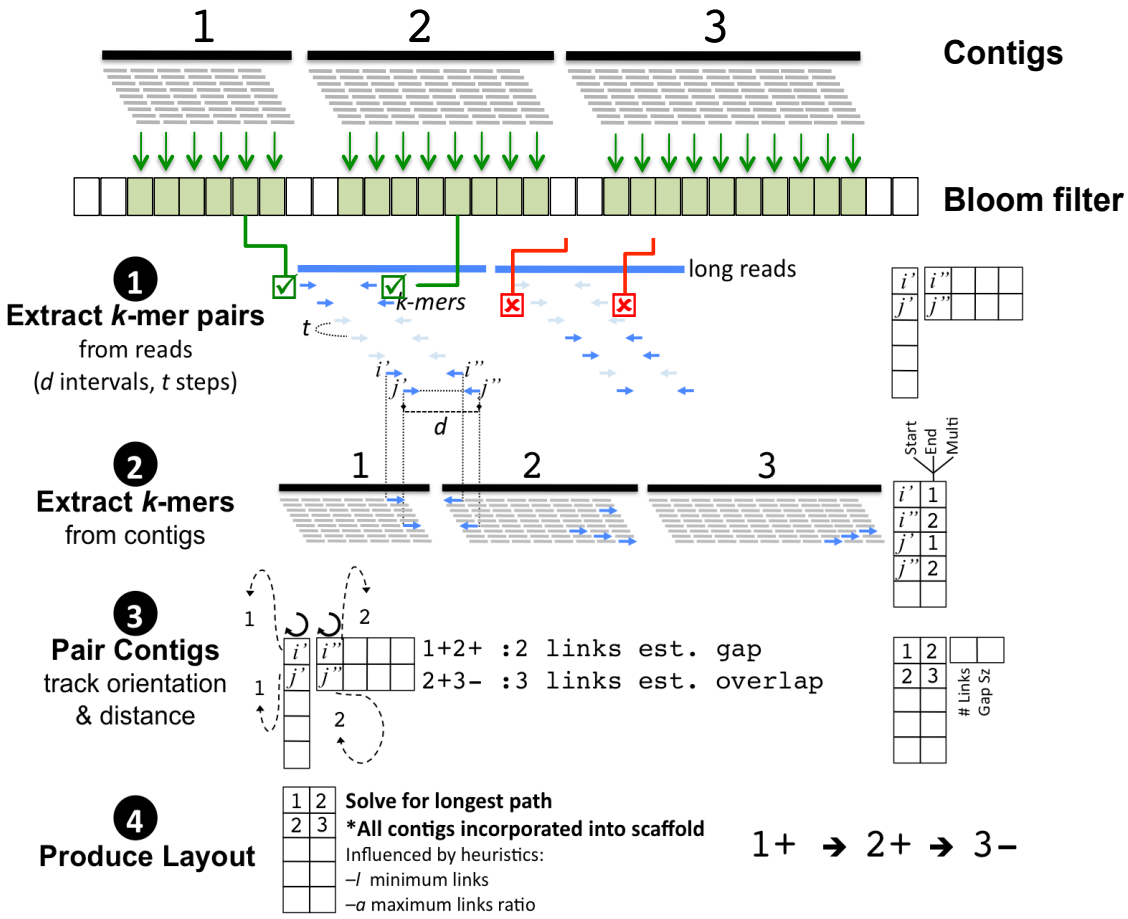

Figure 1. LINKS algorithm. Contigs (three thick black rectangles) are, optionally, shredded into *k*-mers and those *k*-mers used to construct a Bloom filter. Long reads (blue rectangles) are processed and *k*-mer pairs *i*' and *i*'' extracted at an interval corresponding to the input distance (-*d*), and window step (-*t*), but stored in memory (step 1, matrix on the right) only if both *k*-mers of a pair are found in the Bloom filter (**step 1**). Contigs are shredded into *k*-mers once more (**step 2**) using the same *k* value, but stored in memory (step 2, matrix on the right) only when its pair, identified in step 1, exists in memory. In **step 3**, contigs are paired when *k*-mers are not observed in the same sequence. Iterating through the data structure from step 1 and verifying placement (Start, End) and multiplicity (Multi) provides contig linkages, which are stored into memory (step 3, matrix on the right). In **step 4**, the scaffold layout is produced by incorporating all contigs into a scaffold, verifying neighbours and merging only when user-defined parameters support it (>= *l* minimum number of links and <= *a* maximum link ratio between alternate-to-primary linkage).

Consider the following contig pairs (AB, AC and rAD):

```
    A         B
========= ========
  ->       <-
   ->        <-
    ->      <-
       ->       <-


    A       C
========= ======
  ->        <-
    ->        <-


   rA        D           equivalent to rDA, in this order
========= =======
      ->   <-
     ->   <-
       ->   <-
```

Two parameters control scaffolding (-l and -a). The -l option specifies the minimum number of links (read pairs) a valid contig pair MUST have to be considered. The -a option specifies the maximum ratio between the best two contig pairs for a given seed/contig being extended. For example, contig A shares 4 links with B and 2 links with C, in this orientation. contig rA (reverse) also shares 3 links with D. When it's time to extend contig A (with the options -l and -a set to 2 and 0.7, respectively), both contig pairs AB and AC are considered. Since C (second-best) has 2 links and B (best) has 4 (2/4) = 0.5 below the maximum ratio of 0.7, A will be linked with B in the scaffold and C will be kept for another extension. If AC had 3 links the resulting ratio (0.75), above the user-defined maximum 0.7 would have

caused the extension to terminate at A, with both B and C considered for a different scaffold. A maximum links ratio of 1 (not recommended) means that the best two candidate contig pairs have the same number of links -- LINKS will accept the first one since both have a valid gap/overlap. When a scaffold extension is terminated on one side, the scaffold is extended on the "left", by looking for contig pairs that involve the reverse of the seed (in this example, rAD). With AB and AC having 4 and 2 links, respectively and rAD being the only pair on the left, the final scaffolds outputted by LINKS would be:

1) rD-A-B
2) C

LINKS outputs a .scaffolds file with linkage information between contigs (see "Understanding the .scaffolds csv file" below) Accurate scaffolding depends on many factors. Number and nature of repeats in your target sequence, optimum adjustments of distance (-d), deviation on the distance (-e), kmer sizes (-k), Minimum number of links (-l) and link ratio (-a) and data quality will all affect LINKS's ability to build scaffolds.

NOTE: IT IS ADVISED TO RUN LINKS WITH SMALLER DISTANCES (-d) FIRST, ESPECIALLY WHEN ASSEMBLIES ARE FRAGMENTED.

MPET INPUT
==========
In v1.7, a new option (-m) instructs LINKS that the long-read source (-s) is MPET. The users should prepare their input as specified in:
cd test
runme_EcoliK12singleMPET.sh

The MPET input is a custom format akin to FASTA and the sequence record must consist of read1:read2

>template

ACGACACATCTACGCAGCGACGACGATAAATATAC:ATCAGCACAGCGACGCAGCGACAGCAGGACGACGAC

NOTES:

-Paired MPET reads are supplied in their original outward orientation <- ->

-MPET sequences do not need to be trimmed (the Bloom filter will take care of eliminating erroneous kmers not found in the assembly)

-You CANNOT combine MPET and long reads simultaneously in the same LINKS process

-You may trim or process MPET reads if you wish (eg. with NxTrim), but remember to supply resulting MPETs in their original, outward-facing configuration (ie. <- ->). The script in ./tools/makeMPETOutput2EQUALfiles.pl does that for you.

-The default behaviour is to extract kmer pairs from long-read FASTA/FASTQ files specified in -s.

Alternatively, when set to the MPET read length, the -m option will signal LINKS to extract kmer pairs across a distance set in -d, for each MPET pair supplied in files supplied under -s

When doing so, ensure that -t is set to extract at least ~5 kmer pairs/MPET pair.

As a rule of thumb, -l should be set to at least double that value (-l 10 in this case)

Preparing the MPET input

For each fastq MPET file, convert in fasta:

```
 gunzip -c EcMG1_S7_L001_R1_001.fastq.gz | perl -ne
'$ct++;if($ct>4){$ct=1;}print if($ct<3);' > mpet4k_1.fa
 gunzip -c EcMG1_S7_L001_R2_001.fastq.gz | perl -ne
'$ct++;if($ct>4){$ct=1;}print if($ct<3);' > mpet4k_2.fa
```

Generate the paired input (refer to the tools folder):

```
Usage: ./makeMPETOutput2EQUALfiles.pl
<fasta file 1>
<fasta file 2>
<read pair orientation 0/1,  0=raw MPET (<-->)   1=PET (-><-) >
** fasta files must have the same number of records & arranged in the
same order
```

echo mpet4k_1.fa_paired.fa > mpet.fof

OUTPUT FILES

Output files	Description
.log	text file; Logs execution time / errors / pairing stats
.pairing_distribution.csv	comma-separated file; 1st column is the calculated distance for each pair (template) with reads that assembled logically within the same contig. 2nd column is the number of pairs at that distance
.pairing_issues	text file; Lists all pairing issues encountered between contig pairs and illogical/out-of-bounds pairing
.scaffolds	comma-separated file; see below
.scaffolds.fa	FASTA file of the new scaffold sequence
.bloom	Bloom filter created by shredding the -f input into k-mers of size -k
.gv	scaffold graph (for visualizing merges), can be rendered in neato, graphviz, etc
.assembly_correspondence.tsv	correspondence file lists the scaffold ID, contig ID, original_name, #linking kmer pairs, links ratio, gap or overlap
.simplepair_checkpoint.tsv	checkpoint file, contains info to rebuild datastructure for .gv graph
.tigpair_checkpoint.tsv	When -b BASNAME.tigpair_checkpoint.tsv is present, LINKS will skip the kmer pair extraction and contig pairing stages. Delete this file to force LINKS to start at the beginning **This file can be used to:** **-more quickly test certain parameters (-l min. links / -a min. links ratio)** **-quickly recover from crash** **-explore very large kmer spaces**

Interpreting .assembly_correspondence.tsv

This human-readable correspondence file lists the scaffold ID, contig ID, original assembly contig name, contig/sequence orientation, #linking kmer pairs, links ratio, gap or overlap(-) in this order

Interpreting the graph / .gv file

-Vertices correspond to the sequences being considered for scaffolding, with the LINKS re-numbered sequences displayed in each vertex (unlinked sequences are not shown)
-Edges are drawned between vertices when there is evidence for linking scaffolds (even if they are no ultimately scaffolded)
-Only vertices/scaffolds highlighted in blue satisfied user-specified scaffold criteria (l and a parameters and satisfied logic/distance). These are scaffolded in the final LINKS output
-Each edge in the graph will have 3 types of information (l=,g=,type=)
l=:number of kmer pairs linking any two vertices/sequences
g=:estimated gap or overlap (-) length between any two sequences
type=:refers to the orientation of the sequences (forward=1,reverse=0)

Understanding the .scaffolds csv file

scaffold1,7484,f127Z7068k12a0.58m42_f3090z62k7a0.14m76_f1473z354
column 1: a unique scaffold identifier column 2: the sum of all contig sizes that made it to the scaffold/supercontig column 3: a contig chain representing the layout:

e.g. f127Z7068k12a0.58m42_f3090z62k7a0.14m76_f1473z354

It means: contig f127 (strand=f/+), size (z) 7068 (Z if contig was used as the seed sequence) has 12 links (k), link ratio of 0.58 (a) with a mean gap of 42nt (m) with reverse (r) of contig 3090 (size 62) on the right. if m values are negative, it's just that a possible overlap was calculated using the mean distance supplied by the user and the position of the reads flanking

the contig. Negative m values imply that there's a possible overlap between the contigs. But since the pairing distance distribution usually follows a Normal/Gaussian distribution, some distances are expected to be larger than the median size expected/observed. In reality, if the exact size was known between each paired-reads, we wouldn't expect much negative m values unless a break occurred during the contig extension (likely due to base errors/SNPs).

License