Short Sequence Assembly by K-mer search and 3' read Extension (SSAKE)

SSAKE v3.6 Rene L. Warren, 2006-2010
email: rwarren at bcgsc.ca

Description

SSAKE is a genomics application for de novo assembly of millions of very short DNA sequences.
It is an easy-to-use, robust, reliable and tractable clustering algorithm for very short sequence reads, such as those generated by Illumina Ltd.

For best performance

Best assembly results are achieved with quality-trimmed reads. When dealing with Solexa/Illumina sequences, remove low quality bases, whenever possible, with:

```
TQS.py -f _seq.txt -q _prb.txt -t 5 -d 5 -l #CYCLES -c 20  OR OTHER SETTINGS
TQSfastq.py -f myfile.fq -t 20 -c 30 -e 64

example:
~/ssake_v3-6/tools/TQSfastq.py -f ../myIlluminaSeqLane_1.fq -c 30 -t 20  ##mate
no.1
~/ssake_v3-6/tools/TQSfastq.py -f ../myIlluminaSeqLane_2.fq -c 30 -t 20  ##mate
no.2
~/ssake_v3-6/tools/makePairedOutput2UNEQUALfiles.pl
../myIlluminaSeqLane_1.fq_T20C20E64.trim.fa
../myIlluminaSeqLane_2.fq_T20C20E64.trim.fa
will produced "paired.fa" and "unpaired.fa"
~/ssake_v3-6/SSAKE -f paired.fa -g unpaired.fa -p 1 -m 17 -o 4 -c 1
```

The scripts are located in ./tools subdirectory included with this release.
It is recommended that you run TQS.py/TQSfastq.py for every tile (batch job) and cat the outputted fasta file, especially if your data set is large (e.g. entire flowcell)

For trimming paired-end sequences (using _seq.txt and _prb.txt from Illumina), please refer to TRIMMING_PAIRED_READS.README located in the ./tools subdirectory

What's new in v3.6+ ?

v3.6+ supports various insert size sequence libraries. To work with paired data, users must add ":insert_size" (e.g. >SLXA23-1-1-2-13:200 for a 200bp library) at the end of the fasta header (>) for each pairs. v3.6 also has preliminary support for Sanger, paired-end reads.

What's new in v3.5+ ?

In v3.5, the read pairing logic is used in the extension process. More specifically, passed the upper bound insert size, only forward reads AND reverse reads having their assembled mate already assembled in the contig being built will be considered for extension. This has for effect to mitigate contig misassemblies due to repeats. It will also extend the end of adjacent contigs in a scaffold in an effort to fill gaps - the resulting contigs are placed in the .mergedcontigs file.

What's new in v3.4+ ?

Version 3.4 exploits paired-end reads to explore possible contig merges within scaffolds (Consecutive contigs >= -z bases must overlap by -m bases or more). Version v3.4.1 allows a user to merge all contigs of a scaffold by padding predicted gaps with Ns (-n 1) and predicted but undetected overlaps with a single (n). Merged contigs are outputted in the .mergedcontigs file. The default behaviour (-n 0) is to NOT pad the gaps with Ns (v3.4 behaviour). In the v3.4.1, the .readposition file tracks read names instead of read sequences as the latter can be inferred from the start and end coordinates.

What's new in v3.3+ ?

Fixed a bug in PET routine. User can now track read position and individual base coverage for reads *fully embedded* within contigs, using the -c option.

What's new in v3.2.1+ ?

This release runs ~30% faster and requires ~33% less RAM compared to 15-node prefix tree SSAKE v3.2.0.1 beta.
Compared to the previous v3.2 release (11-node prefix tree), it will run at ca. double the speed, requiring ~20% more RAM.

What's new in v3.2+ ?

The -t option, first introduced in SSAKE 1.3, is back in this release. The option allows you to trim your contigs in 3', 1 base at a time until a maximum base trim value (-t) is reached. This option yields longer contigs, but increases assembly run time and, at high t values, might introduce contig misassemblies if your run parameters (i.e. -m, -o & -r) are not stringent enough. At -m 16 -o 3 -r 0.7, best results were obtained with -t 1. That's because it removes bases that cause premature breaks during the fragment assembly. If set, end-trimming kicks in only when all possibilities have been exhausted for a contig extension.

This release also fixes a major bug that prevented SSAKE from exploring the entire read space for contig extensions seeded by the shorter reads.

What's new in v3.1+ ?

SSAKE now allows users to input a fasta file with DNA sequences for use as seeds to nucleate contig extension.
This feature can be used to extend existing/known DNA sequences using millions of short reads.
There's a new input format for paired-end reads, which allows reads of variable length to be considered (such as quality-trimmed reads)

What's new in v3.0+ ?

SSAKE supports Illumina paired-end read data to build scaffolds.

What's new in v2.0+ ?

SSAKE can now handle error-rich data sets by looking through the overlapping k-mer space for consensus bases overhanging a seed sequence or contig.
SSAKE now runs on reads of various lengths. That means quality base trimming of individual sequences can be achieved (using TQS.py/TQSfastq.py supplied in ./tools directory).

Implementation and requirements

SSAKE is implemented in perl and runs on linux.
Error handling by deriving an overhang consensus from overlapping reads is an original idea by William Jeck & Josie Reinhardt (VCAKE v1.0 William Jeck, May 2007) and implemented in SSAKE from scratch.

Side-by-side comparison between ssake2.0 and vcake1.0 indicates that SSAKE is nearly 3-fold faster and yields contigs that are as contiguous and accurate.

The python version 2.0 (released in ssake_v2.0.tar.gz and distributed under ./tools) has not yet been fully tested.
Due to SSAKE's memory requirements, you would need a version of the perl interpreter compiled for 64-bit computers if you intend to cluster millions of short sequences.
Development of SSAKE was done using perl v5.8.5 built for x86_64-linux-thread-multi

With ssake_3.2.pl -p 0
You can cluster ~5 million 25-mers with SSAKE on a computer with 4GB RAM
You can cluster 60-80 million 25-mers with SSAKE on a computer with 32GB RAM

PLEASE READ:
*When using paired-end reads (-p 1), SSAKE tracks in memory all paired reads located in contigs >= z. That means that the memory usage will increase drastically with the size of your data set. Just be aware of this limitation and don't be surprised if you observe a lot of data swapping to disk if you attempt to run SSAKE on a machine with little RAM.

SSAKE might not be suited to work with 454-type reads. Simply because recurring base insertions/deletions errors, such as those commonly seen in homopolymeric regions, will not cluster well in the context of the SSAKE algorithm scheme. Sanger reads are ok, as long as reads are quality-trimmed.

Install

Download the tar ball, gunzip and extract the files on your system using:

```
gunzip ssake_v3-6-tar.gz
tar -xvf ssake_v3-6-tar
```

Change the shebang line of SSAKE to point to the version of perl installed on your system and you're good to go.

Documentation

Refer to the SSAKE.readme file on how to run SSAKE and the SSAKE web site for information about the software and its performance
www.bcgsc.ca/bioinfo/software/ssake

Questions or comments? We would love to hear from you!

Citing SSAKE

Thank you for using, developing and promoting this free software.
If you use SSAKE for you research, please cite:

Warren RL, Sutton GG, Jones SJM, Holt RA. 2007. Assembling millions of short DNA sequences using SSAKE. Bioinformatics. 23(4):500-501

Running SSAKE

e.g. SSAKE -f paired.fa -m 17 -o 4 -r 0.7 -p 1 -c 1 -e 0.75 -k 2 -a 0.6 -z 50 -g unpaired.fa

Usage: ./SSAKE [v3.6]
-f File containing all the [paired (-p 1) / unpaired (-p 0)] reads (required)
! Target insert size must be indicated at the end of the header line (e.g. :200 for a 200bp insert)

```
      ! Paired reads must be separated by ":"
        >template_name:200
        ACGACACTATGCATAAGCAGACGAGCAGCGACGCAGCACG:GCGCACGACGCAGCACAGCAGCAGACGAC
-s  Fasta file containing sequences to use as seeds exclusively (specify only if
different from read set, optional)
-m  Minimum number of overlapping bases with the seed/contig during overhang
consensus build up (default -m 15)
-o  Minimum number of reads needed to call a base during an extension (default -
o 2)
-r  Minimum base ratio used to accept a overhang consensus base (default -r 0.7)
-t  Trim up to -t base(s) on the contig end when all possibilities have been
exhausted for an extension (default -t 0, optional)
-c  Track base coverage and read position for each contig (default -c 0,
optional)
-b  Base name for your output files (optional)
-z  Minimum contig size to track base coverage and read position (default -z
100, optional)
-p  Paired-end reads used? (-p 1=yes, -p 0=no, default -p 0, optional)
-v  Runs in verbose mode (-v 1=yes, -v 0=no, default -v 0, optional)
============ Options below only considered with -p 1 ============
-e  Error (%) allowed on mean distance   e.g. -e 0.75  == distance +/- 75%
(default -e 0.75, optional)
-k  Minimum number of links (read pairs) to compute scaffold (default -k 4,
optional)
-a  Maximum link ratio between two best contig pairs *higher values lead to
least accurate scaffolding* (default -a 0.5, optional)
-x  Minimum overlap required between contigs to merge adjacent contigs in a
scaffold (default -x 20, optional)
-n  N-pad gaps (-n 1=yes, -n 0=no, default -n 0, optional)
-g  Fasta file containing unpaired sequence reads (optional)


Test data
---------

Go to ./test

run ../SSAKE -f Herpesvirus_3.60kb.reads.fa -m 16 -o 2 -r 0.6 -p 0 -t 0 -c 1 -b
myFirstSSAKErun
compare your results with Herpesvirus_3.60kb.reference.fa
** This error-rich, simulated data set was made available as part of the VCAKE
v1.0 distribution
it represents a 60,000 bp stretch of the NC_001348.1 human herpes virus 3 **

run ../SSAKE -f Herpesvirus_3.60kb.reads.fa -m 16 -o 2 -r 0.6 -p 0 -s
Herpesvirus_3.60kb.seed.fa -t 0
The assembly above uses a single seed sequence located in
(Herpesvirus_3.60kb.seed.fa)
compare your results with Herpesvirus_3.60kb.reference.fa to see how successful
the nucleation was
```

How it works
---

1. Sequence Overlap

Short DNA sequences of length l in a single multi fasta file -f are read in memory, populating a hash table keyed by unique sequence reads with pairing values representing the number of sequence occurrence in the input read set. The normalized sequence reads are sorted by decreasing abundance (number of times the sequence is repeated) to reflect coverage and minimize extension of reads containing sequencing errors. Reads having sequencing errors are more likely to be unique in the entire read set when compared to their error-free counterparts. Sequence assembly is initiated by generating the longest 3'-most word (k-mer) from the unassembled read u that is shorter than the sequence read length l. Every possible 3' most k-mers will be generated from u and used in turn for the search until the word length is smaller than a user-defined minimum, m. Meanwhile, all perfectly overlapping reads will be collected in an array and further considered for 3' extension once the k-mer search is done. At the same time, a hash table c will store every base along with a coverage count for every position of the overhang (or stretches of bases hanging off the seed sequence u).

Once the search complete, a consensus sequence is derived from the hash table c, taking the most represented base at each position of the overhang. To be considered for the consensus, each base has to be covered by user-defined -o (set to 2 by default). If there's a tie (two bases at a specific position have the same coverage count), the prominent base is below a user-defined ratio r, the coverage -o is to low or the end of the overhang is reached, the consensus extension terminates and the consensus overhang joined to the seed sequence/contig. All reads overlapping are searched against the newly formed sequence and, if found, are removed from the hash table and prefix tree. If they are not part of the consensus, they will be used to seed/extend other contigs, if applicable. If no overlapping reads match the newly formed contig, the extension is terminated from that end and SSAKE resumes with a new seed. That prevents infinite looping through low-complexity DNA sequences. In the former case, the extension resumes using the new [l-m] space to search for joining k-mers.

The process of progressively cycling through longer to shorter 3'-most k-mer is repeated after every sequence extension until nothing else can be done on that side. Since only left-most searches are possible with a prefix tree, when all possibilities have been exhausted for the 3' extension, the complementary strand of the contiguous sequence generated is used to extend the contig on the 5' end. The DNA prefix tree is used to limit the search space by segregating sequence reads and their reverse-complemented counterparts by their first eleven 5' end bases.

There are three ways to control the stringency in SSAKE:
1) Disallow read/contig extension if the coverage is too low (-o). Higher -o values lead to shorter contigs, but minimizes sequence misassemblies.
2) Adjust the minimum overlap -m allowed between the seed/contig and short sequence reads. Higher m values lead to more accurate contigs at the cost of decreased contiguity.
3) Set the minimum base ratio -r to higher values

2. Building scaffolds with SSAKE

If the -p option is set to 1, it is assumed that the data supplied in the fasta file (-f) consists of paired-end reads, concatenated together on the same line, but separated by ":" -- see "Input sequences" section below.

During data input, pairs are split and both used to fill the prefix tree and hash table, as described in Warren et al. 2007.
With the -p option set, the position of all sequence reads in contigs -z and larger are tracked.
If a file is specified with -g, its unpaired sequences will be co-assembled along with paired reads during the SSAKE 3' extension but the former reads will NOT be tracked.

At the end of the overlap phase (aka contig extension), the -f fasta file is read again, associating reads with their pairs.
For each read pairs, putative contig pairs (pre-scaffolding stage) are tallied based on the position/location of the paired-end reads on different contigs. Contig pairs are only considered if the calculated distance between them satisfy the mean distance specified (>template:insert_size) while allowing for a deviation (-e), also defined by the user. Only contig pairs having a valid gap or overlap are allowed to proceed to the scaffolding stage.
Please note that this stage accepts redundancy of contig pairs (i.e. a given contig may link to multiple contigs, and the number of links (spanning pairs) between any given contig pair is recorded, along with a mean putative gap or overlap(-)).
Once pairing between contigs is complete, the scaffolds are built using contigs (-z or larger) as seeds. Every contig is used in turn until all have been incorporated into a scaffold.

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 (-k and -a). The -k 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 -k 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 -- SSAKE 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 SSAKE would be:

1) rD-A-B
2) C

SSAKE 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 insert_size, -e, -k and -a and data quality/size of sequence set (more doesn't mean better) will all affect SSAKE's ability to build scaffolds.

3. Using a seed sequence file

If the -s option is set and points to a valid fasta file, the DNA sequences comprised in that file will populate the hash table and be used exclusively as seeds to nucleate contig extensions (they will not be utilized to build the prefix tree). In that scheme, every unique seed will be used in turn to nucleate an extension, using short reads found in the tree (specified in -f). This feature might be useful if you already have characterized sequences & want to increase their length using short reads. That said, since the short reads are not used as seeds when -s is set, they will not cluster to one another WITHOUT a seed sequence file. Also, to speed up the assembly, no imbedded reads (i.e. those aligning to the seed in their entirety) are considered. Only reads that contribute to extending a seed sequence are noted.

When -s is set, the .contigs file lists all extended seeds, even if it's by a single base. The .singlets will ONLY list seeds that could not be extended. Unassembled microreads will NOT be outputted.

*Refer to the "Test data" section below for a concrete example

4. Using seeds (-s) with mate pairs (-p 1):

If more than one seed is supplied in the -s file and you're providing paired-end reads (-p 1), SSAKE will attempt to scaffold extended seeds (if a seed wasn't extended it will end-up in the singlets and will not be considered for scaffolding) using the supplied mate pairs.

Input sequences

UNPAIRED:

DNA sequences can be in lower caps as well

```
>PX1CG_29
TTAACACTTTCGGATATTTCTGATG
>PX1CG_35
CTTTCGGATATTTCTGATGAGTCGA
>PX1CG_64
TTATCTTGATAAAGCAGGAATTACT
...
```

PAIRED:

```
>2-1-464-197:200
TGGCTCACCCCTGTAATCCCAGCACT:CTCCCAGGTTCAAGCGATTCTCCTGC
>2-1-783-425:300
GTCTGAGGGTCCCAGGAACCAG:TGCCCCAGAGGTGGGAGCAGGGGA
>2-1-662-655:1000
TGAATCCCCACCAGGCGCCTTCGG:CACTTTATTATTAATGTACAAAAT
```

-Paired sequences must be concatenated together in one fasta-like entry, separated by ":". For example, TGGCTCACCCCTGTAATCCCAGCACT:CTCCCAGGTTCAAGCGATTCTCCTGC consists of two paired reads. Changes to the input was made to allow reads of variable length (e.g. quality-trimmed reads) to be considered by SSAKE. As of v3-6, the header line [>] must have [:insert_size] at the very end (see above example)

-The -f option can read either paired or unpaired sequences, depending whether -p is set or not, respectively.
Users can co-assemble paired and unpaired reads if they wish. If so, the unpaired reads are inputted using the -g option.

General points:
-To be considered, sequences have to be longer than 16 nt or -m (but can be of different lengths). If they are shorter, the program will simply omit them from the assembly and will be placed in the .shorts file
-Short sequences that have not been extended are placed in the .singlets file
-As before, the length of individual sequence is used to determine the size of the right-most subsequence to look for initially
-Reads containing ambiguous bases "." and characters other than ACGT will be ignored entirely
-Spaces in fasta file are NOT permitted and will either not be considered or result in execution failure

Output files

.contigs   :: fasta file; All sequence contigs
.log   :: text file; Logs execution time / errors / pairing stats (if -p is set to 1)
.short   :: text file; Lists sequence reads shorter than a set, acceptable, minimum
.singlets :: fasta file; All unassembled sequence reads

-p 1
.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
.mergedcontigs :: fasta file; All merged/unmerged contigs >= -z bases within scaffolds are listed. The overlap sequence between contigs (>= -x bases) will be shown in lower case within the merged contig. Note that *perfect* sequence overlap has to occur between 2 predicted adjacent contigs of a scaffold in order to merge. It is possible that two contigs merge even though they are NOT predicted to do so (perhaps because insert size range supplied is off or mate pairs are misassembled). When two consecutive contigs do not physically overlap and the -n option is set to 1, then gaps will be padded with Ns of length corresponding to the predicted gap size m (refer to Understanding the .scaffolds csv file below) and predicted but undetected overlaps with a single (n).

-c 1 (WARNING: ASSOCIATED FILES CAN BECOME VERY LARGE!)
.readposition   :: this is a text file listing all whole (fully embedded) reads, start and end coordinate onto the contig (in this order). For reads aligning on the minus strand, end coordinate is < start coordinate
.coverage.csv   :: this is a comma separated values file showing the base coverage at every position for any given contig   > -z

Understanding the .contigs fasta header
----

e.g.
>contig27|size52|read193|cov92.79

contig id# = 27
size (G) = 52 nt
number of reads (N) = 193
cov [coverage] (C) = 92.79

the coverage (C) is calculated using the total number (T) of consensus bases [sum(L)] provided by the assembled sequences divided by the contig size:

C = T / G

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

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).

Use makeFastaFileFromScaffolds.pl included in this distribution to make a scaffold fasta file (ordered and oriented contig sequences) using the layout recipe (contig chain) shown above.

Understanding the .coverage.csv file

e.g.

```
>contig1|size60000|read74001|cov37.00
12,12,13,13,13,14,14,15,16,16,20,21,22,23,25,26,27,28,27 ...
```

Each number represents the number of reads covering that base at that position.

Understanding the .readposition file

e.g.

```
>contig1|size60000|read74001|cov37.00
READ_85952,3,32
READ_92647,6,35
READ_72602,8,37
READ_29659,9,38
READ_74582,11,40
READ_97793,11,40
READ_85742,11,40
READ_95375,12,41
READ_9721,15,44
READ_49141,16,45
READ_43328,18,1
```

READ_94449,18,1

In this order: read name [template th -p 1 :: name followed with 1 or 2, corresponds to the order in the sequence input (1:2)], start coordinate, end coordinate. end < start indicates read is on minus strand

SSAKE does not

-Take into consideration base quality scores. It is up to the user to process the sequence data before clustering with SSAKE.
      Python scripts (TQS.py, TQSfastq.py, TQSexport.fq) are provided to help trim poor quality bases off Illumina sequences.
      Refer to TQS.readme and TRIMMING_PAIRED_READS.README included in this distribution (in the ./tools subdirectory) for information on how to run those programs
-Consider sequence read having any character other than A,C,G,T and will skip these reads entirely while reading the fasta file.

License

Enjoy SSAKE responsibly!