

SL Finder Manual

Index

- 1) Introduction --- pag 1
- 2) Citation --- pag 2
- 3) Download and Installation --- pag 2
- 4) Running SL Finder - Quick version --- pag 2
- 5) The pipeline --- pag 4
 - Strategy, basic assumptions and problems to circumvent --- pag 4
 - Pipeline implementation --- pag 7
 - SLFinder-step0 --- pag 8
 - SLFinder-step1 --- pag 9
 - SLFinder-step2 --- pag 12
 - SLFinder-step3 --- pag 13
- 6) Warning files --- pag 18
- 7) Working with the pSL sequences --- pag 20
- 8) How to... --- pag 20

Introduction

SL Finder is a four step pipeline of bioinformatic analyses meant to identify potential Spliced Leader (pSL) sequences from Transcriptome data assembled *de novo*, without making use of known SL sequences and with minimal manual curation by the user:

- SLFinder-step0: Filters a Trinity assemblies and retrieves the firsts and last XXpb of the longer isoform of each assembled gene. According to Trinity contig naming convention.
- SLFinder-step1: Creates and evaluates “Hook” sequences that more likely represent SLs and retrieves contig regions, either in the 3’ or 5’ region for further analyses.
- SLFinder-step2: Aligns the sequences identified for each hook and automatically trims them to generate the more likely pSL sequences.
- SLFinder-step3: Blast the pSL sequences against a reference transcriptome looking to verify their sequence and identify their coding location and copy number and identify the transcripts with validated pSL variants. If provided, it uses an external reference to annotate the genomic region with each pSLs and discard those with matches.

Citation

Please cite...

Download and Installation

Currently the pipeline has been tested on a Linux environment. The scripts can be downloaded from XXXXXXXXXXXXXXXXXXXX and require execution permissions to work properly. These can be granted with the following command:

```
chmod +x SLFinder-step*
```

Additionally, the following programs and/or packages installed and added to the PATH:

Ncbi-blast:

https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download

Cd-Hit: <https://github.com/weizhongli/cdhit>

Jellyfish: <https://github.com/gmarcais/Jellyfish>

MAFFT: <https://mafft.cbrc.jp/alignment/software/source.html>

Salmon: <https://github.com/COMBINE-lab/salmon>

Seqkit: <https://github.com/shenwei356/seqkit>

Trinity: <https://github.com/trinityrnaseq/trinityrnaseq/wiki>

Trimal: <http://trimal.cgenomics.org/trimal>

Weblogos: <http://weblogo.berkeley.edu/>

Please follow each program installation instructions before running this pipeline. In addition, programs commonly included as part of the Linux installation are also used for file manipulation and some calculations like awk, cut, grep, sed, split and wc.

Running SL Finder - Quick version

SL Finder is designed to work with several assemblies of the same species. For this end each assembly should be named with a distinct ID followed by an extension that identifies it as an assembly (ej: [Sample]_assembly.fasta) and all assemblies must be in the working directory. It is important that the assembly to be conducted following a de novo approach with no read normalization. Using a reference for transcript assembly is likely to exclude the SL from the transcripts as the reads fail to properly map to them, while a

normalization strategy is likely to reduce the number of reads with SLs sequences since they are expected to be overrepresented when compared with a random sequence. Since Trinity is not only a popular de novo assembler but the pipeline requires it to work, we will assume to assemble your transcriptomes. Use the following command for the assemblies:

```
Trinity --seqType fq --left reads_1.fq --right reads_2.fq --CPU X
--max_memory XXG --no_normalize_reads
```

In addition: this pipeline requires the species genome (or as close related one as possible) and a reference for protein annotation like Swiss-Prot from Uniprot.

Once the assemblies and references are ready follow each step of the pipeline:

1) Filter assemblies – step0:

```
SLFinder-step0 -a “_assembly.fasta” -f “_filtered.fasta” -t n°
```

Here -a is the extension for the assemblies and -f the extension for the filtered files. Internally the pipeline runs “ls *_assembly.fasta” in the working directory to get the file name list. By default -a is “_Trinity.fasta” and -f “_Trinity.filtered”. This script might take several minutes to a few hours, to improve running times assign several processors to run in parallel with “-t” parameter. These are used by seqkit grep to speed up the retrieval of the longest isoforms. Other programs used in the pipeline latter on also allow for parallelization like BLAST, CD-HIT-EST or MAFFT.

Lastly, if you are using a different assembler than trinity or want to follow a different filtering strategy you can replace this step by simple introduced your custom filtered files to the next step.

2) Produce Hooks and retrieve matching sequences – Step1:

```
SLFinder-step1 -a “_assembly.fasta” -f “_filtered.fasta” -t n°
```

Details of how this script works are explained below, but to work the script needs again the original assemblies as well the filtered ones. The latter are used to generate the Hooks representing the pSL, then a BLAST search is carried out to identify matching regions in the original assembly. Results are filtered according to the consistence of their orientation in the contigs and relative frequency, then the sequence matches of the more promising Hooks is retrieved for further analyses.

This script will also generate an output folder to store the results of the entire pipeline (named by Default “SL-analysis”), with three folders inside: Results, Warnings and

temp_files. Inside Results there will be three files with the hook information: “Step1-All_Hooks”, “Step1-Best_Candidates” and “Best-Hook.fasta”. As a safety measure, this script only runs if the output folder doesn’t already exist but bear in mind that the following steps delete the results of their previous run stored in this folder.

3) Trimm Hook matches – Step2

SLFinder-step2 -t n°

This step only works with the files generated during Step1 and doesn’t require additional inputs. However, this part of the pipeline is kept separated because several Trimal parameters might need fine-tuned to your specific dataset, and in some cases require manual curation. Basic information for the Trimal Fasta files is given in the file Step2_Trimal-Stats and unreliable alignments listed on “Step2_Manual-curation”. To facilitate any manual curation, a sequence logos of each alignment is generated and stored on “Analysis directory”/Results/weblogos/alignments

Lastly, all Trimal filtered sequences are combined and clustered according sequence identity before been used in the next step.

4) Localize pSL sequences in a reference genome – Step3

SLFinder-step3 -g [Reference Genome] -br “external_reference” -t n°

The pSL sequences obtained so far are then located in a reference Genome by a BLAST search (with no gaps allowed and a 100% identity cutoff) to verify their sequence and identify potential donor sites (GT) at the 3’ of the match or their reversed complement at the 5’ (AC). Additionally, an external reference (provided by the command “-br”) is used to annotate the loci encoding the pSL. Results are summarized in “Step3_Loci.tab” and the validated pSL sequences in SL.fa

The pipeline

Strategy, basic assumptions and problems to circumvent

While not requiring known SL sequences to work, this pipeline makes some basic assumptions about the SL mechanism:

1. The sequence is located in the 5’ end of the transcript
2. It is present on the transcripts of many genes

3. It is not a palindrome who can be mapped to the same location in both possible orientations
4. There is at most one copy of it on each transcript
5. When mapped to the genome there is a canonical Splicing donor site at the 3'.

In addition, some features and limitations of the technology commonly used to assemble transcriptomes from Next Generation Sequencing data complicate using a simple pattern search to identify new SL sequences. For example, Poly-A capture used in many protocols to enrich RNA samples with eukaryotic mRNAs has the drawback of generating a lower read coverage toward the 5' region of the assembled transcript (where the SL is located). Another factor to consider is that several assemblers like Trinity can recover several isoforms for the same gene, something great when studying the biology of an organism but that for our purposes it's a source of noise if not properly filtered. And as an extra complication, unless a special protocol is used the strand of each assembled transcript is lost; meaning that we don't know a priori if we assembled the transcript or its reverse complement.

For this reason, the first step is to filter the transcriptome to reduce redundancy. In this implementation this is done based on Trinity cluster information, but other strategies are viable. Next, we generate "Hook" sequences that represent the more common sequences in the filtered assembly by first counting and filtering out those present less than a user given threshold how many times a given string or bases (kmer) is present on filtered assembly; then assemble them in larger sequences using the module Inchworm of Trinity. By using a kmer size slightly less than the SL length its possible to retrieve information from transcript only partially assembled, with a limited compromise on specificity. Transcripts with matching these hooks are then identified in the unfiltered transcriptome by a BLAST search and their location and orientation used to filter likely false positives.

Two filters are currently implemented: Concordance Index (ci) and a frequency filter.

- Concordance Index

Here is when the assumption of the SL not been palindromic comes into play. We assume that if a Hook truly is an SL, it will be oriented in a consistent way in one of these two configurations:

- 1) The assembled hook is a SL and is oriented from 5' to 3' if located at the beginning of the contig, or 3' to 5' if found at the end (Fig. 1), meaning that the reverse complement of the transcript was assembled but the Hook is in the correct orientation.



Fig 1: Expected disposition of a Hook for a true SL.

- 2) The assembled hook is the reverse complement of the SL 5' and is oriented from 3' to 5' if located at the beginning of the contig, or 5' to 3' if found at the end (Fig. 2), meaning that the reverse complement of the transcript, and so is the Hook.



Fig 2: Expected disposition of a Hook for a true SL, but the produced hook is the reverse complement of the SL.

The ci consist in the following formula:

$$ci = |((Sf + Er) - (Sr + Ef)) \div (Sf + Er + Sr + Ef)|$$

Here “Sf” is are the number of times the hook in question has a hit at the start of the coting in the same sense as it was assembled and “Er” is the how many times it was observed at the end in the reverse respective to it, “Sr” and “Ef” their counter parts: In reverse at the start and forward at the end. If the Hook is in fact an SL most if not all hits on the transcriptome are going to be in either one or the other configuration, so the result of the formula will be close to 1.

- Observation count threshold

Testing indicates that, while effective, a high ci filter tends to pass hooks with few matches and whose sequence do not represent SL sequences. For this reason, we include a second filter that discards Hooks with less than the median of observations counts of all other Hooks.

Note1: Transcripts with multiple hits for the same Hook are not considered.

Once filtered, the next step is to recover the correct sequence of the SLs. Its common that that Hooks representing a true SL include more pb on both 5' and 3' ends. This happens both because of the limited information sequences used during their assembly

or because several transcripts begin with the same sequence (i.e. a start codon ATG). Other anomalies include chimeric versions among different SLs, multiple hooks for the same SL or missing bases in the 3' end (usually when there are two or more SL variants that are different on their 5' but the 3' is conserved). This is solved by retrieving the sequence of all the Best Hooks Matches, align them with MAFFT and trimming them automatically with Trimal (or manually if required). While not necessarily eliminating the problem (e.g. if 30 of the 50 transcripts with a true SL have a AT following, these pb are likely to remain in the sequence after the trimming) this process minimizes it.

After trimming, somewhere between dozens or several hundred of very similar sequences (referred as Hook variants) will be produced. They represent all distinct possible SLs sequences present on the transcriptome, unfortunately most of them are likely caused for sequencing errors. The next step is to use a reference genome to localize encoding loci via a BLAST search (no gaps, 100% identity and over an user defined threshold of the pSL coverage). Once the Loci are identified the script searched for potential donor sites ("GT" or "AC" depending on the strand) and the entire region annotated by another BLAST search against an appropriate external reference (e.g. Uniprot). The pSL Loci is considered valid if there is a potential donor sites and the BLAST search brings no results.

Pipeline implementation

Each step in the Pipeline is summarized in Fig 3. Now we will explain how each step is implemented on each step and the options available to change its behavior by the user.

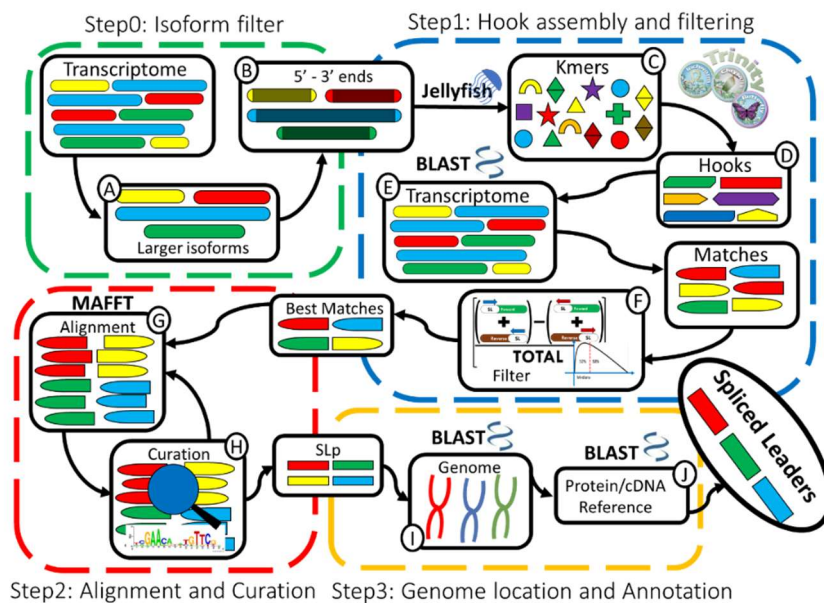


Fig. 3: schematic of the SL Finder pipeline as currently implemented: A) Longer isoform selection; B) Terminal region retrieval; C) Count and filter of Kmers; D) Hook

assembly; E) BLAST search Hooks – Full Transcriptome; F) Hook Filter based on matches orientation and abundance, and sequence retrieval; G) Best Matches sequence alignment; H) Alignment automatic trimming and curation; I) BLAST search pSL – Reference Genome; J) BLAST search pSL Loci – Protein/cDNA reference.

SLFinder-step0

The objective of this script is to filter a transcriptome assembled *de novo* with Trinity. The longer isoform for each gene is retrieved based on the contigs IDs (Fig. 4). Not because we expect that longer isoforms was produced by SL trans-splicing (on the contrary) but to account for incomplete transcript assembly when using Poly-A capture. Once identified, their full sequences are retrieved using “seqkit grep” and the terminal regions with “seqkit subseq”.

Three output files are generated on the working folder: [Sample]_LongerIsoforms-ID with the IDs of the longer isoforms, [Sample]-LongerIsoforms.fa with the complete sequence of the longer isoforms and a third with a user given extension (-f parameter) with the 5’ and 3’ ends of the longer isoforms.

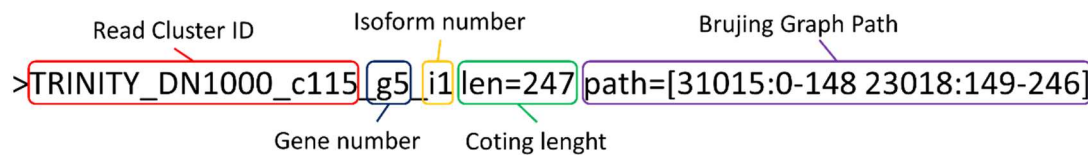


Fig. 4: Contig name convention used by Trinity.

The behavior of Step0 can be customized to a degree with the following parameters:

- a, --assemblies: Extension string identifying the assemblies to be analyzed. As the script is designed to work on Trinity assemblies. By default is “_Trinity.fasta”
- f, --filtered: Extension string identifying the filtered assemblies (5’ and 3’ ends of the longer isoforms). By default is “_Trinity.filtered”
- n, --idnum: This is the maximum number of contig IDs to retrieve from the assembly at a given time with seqkit grep. While efficient, this program can give an out of memory error if given too many entries to search are given. So, the [Sample]_LongerIsoforms-ID is sub divided by lines and the sequences retrieved part by part. Default value is 50,000
- s, --seqcut: Sequence length to retrieve from each contig end, default value is 50pb.
- t, --threads: Number of threads to use when parallelization is possible (seqkit grep in this script), default value is 1.

This step is numbered zero because this part can be easily replaced by other strategies that serve the same end. For example, the sequences of both ends for all contigs can be retrieved and clustered using CD-HIT-EST or the transcriptome can be annotated previously and only one of the transcripts for each gene considered according other criteria. If for some reason a different strategy would work better on your data, you can still use the rest of this pipeline (see below).

SLFinder-step1

This script takes the filtered transcriptome and generates “Hook” sequences that are used to “fish out” common sequences on the original assembly, among which should be the SLs if present on the transcriptome using a BLAST search. In order to work, the script must receive a string identifying the original assemblies and associated filtered sequences (options “-a” and “-f” in the previous script and this one).

-a, --assemblies: Extension string identifying the assemblies to be analyzed. As the script is designed to work on Trinity assemblies. By default is “_Trinity.fasta”

-f, --filtered: Extension string identifying the filtered assemblies (5’ and 3’ ends of the longer isoforms). By default is “_Trinity.filtered”

First, the Kmer present on the filtered sequences are counted with “jellyfish count -C” and those with less than a user given cutoff counts discarded and converted in a Fasta file. To preserve frequency information in the next step, each Kmer sequence is duplicated as many times as it was observed before been assembled by Inchworm. The behavior of this process can be controlled with the following parameters:

-kc, --kmercount: Minimal Kmer observations required to be analyzed. It must be a float number representing the percentage of transcripts with the kmer compared (Default value is 0.0005). Increasing this number will increase the strictness of the analyses but it will reduce the ability to identify low frequency SLs. Also testing shows that increasing this value above 0.001 can greatly reduce the detection power of the pipeline even when the SL Tran-Spliced transcripts are abundant but there is biological diversity in their sequence (i.e. SL-2 in *Caenorhabditis elegans*).

-kl, --kmerlength: Kmer length to count. Due to Inchworm limitations It must be an integer bigger than 15 and smaller than the sequence length analyzed. Longer Kmers will greatly reduce false positives, however they impose a hard limit on the length of the SL that can be detected. Default value is 20.

-ki, --inchwormk: Kmer size used by Inchworm to conduct the assembly. This value must be between 10 and 32, and also be smaller than the value provided with “-kl”. Smaller numbers might lead to chimeric hooks that do not match the original transcriptome while longer values create a larger number of redundant hooks. Default value is 15.

-m, --hookmin: Minimal hook length to analyze. This serves to discard Hooks composed of very few Kmers and simplify results. Default value is 20.

-c, --cdhitH: Sequence identity threshold used for Hook filtration by clustering sequences with CD-HIT-EST. As stated before, it's common that several hooks are created for the same sequence. This option allows to filter these hooks early on the pipeline and reduce running time. Default value is 1 (100% sequence identity).

For simplicity, Hooks keep the ID given by Inchworm but the “;” is replaced by “-” to prevent issues with special characters. Once assembled, the script searches the location of all hooks in the unfiltered transcriptome (necessary for Hook filtering and sequence retrieval) by a BLAST search “blastn -task blastn-short”. It's possible to change the stringency of the BLAST with the following parameters. Since the Hook sequence might include pb that aren't present in any of the transcripts a 100% identity match is not advised. Also, a more permissive configuration allows to detect SL variants that are not represented in the hooks but are similar in sequence (i.e SL2 variants in *C. elegans*).

-bi, --blastnl: Identity threshold used to identify Hook matches in the unfiltered transcriptome. Default value is 90 (2 mismatches in a 20pb Match)

-ba, --blastnA: Minimum hook Match length. Default value is 15.

The results are then filtered according to the ci and relative frequency by the Observation count threshold. These can be controlled with the parameters:

-i, --iorent: Float number representing the cutoff threshold for the ci. Testing shows that Hooks from true SLs have a value close to 1, but it's not uncommon for them to have few instances of transcripts having matches in unexpected orientations so a value of 1 is not advised. Default value of 0.95

-me, --medianFilter hooks according to the median value of their matches count: T (True) or F (False). If this is set to “F” the Observation count threshold is set to 1. Default value is “F”.

Once completed, a blast search of hooks passing the filters against themselves (100% Identity cutoff) in order to detect anomalies as redundancy on sequence or internal

repeated sequences. And sequences matching all best hooks is retrieved to be used for SLFinder-step2. To prevent issues during the next step, the sequences are separated by the Match location and orientation and named: [HookID]-[contig end]-[match orientation]-sequence (e.g. a1-4907-3prima-forward-sequence). This division helps to reduce running times during the sequence alignment in Step 2 and prevents trying to align the same sequence on forward and reverse strands.

To compensate the potential partial recovery of SLs mentioned earlier, three additional pb are included toward the center of the contig. Another complication that can happen in some assemblies is the presence of non-conserved sequence toward the 5' of the assembled transcripts, even when a SL sequence is observed, that likely doesn't have a biological origin. These can greatly increase running time and be detrimental during SLFinder-step2. Our solution is to trim the raw matches using "seqkit subseq", keeping XX pb counted from the more central match coordinate (the presumed 3' end the SL). The sequence length to conserve can be controlled with:

-s, --seqtrimm: Trimm Raw matches larger than X, counting from more central position of the match. Default value is 50

Other important parameters to consider are:

-o, --outdir: Directory where to store the output files, its important that no directory with this name exists. The default name is "SL-analysis"

-M, --Tmaxmemory: Maximum suggested max memory to use by Trinity in Gigabytes
Default value is 1

-t, --threads: Number of threads to use when parallelization is possible (BLAST, Jellyfish, and Trinity in this script), default value is 1.

Main results are provided in four files in Results:

- 1) Step1-All_Hooks: Table with the matches counts for all Hooks on each location and orientation. It includes the hooks, n° observations at the start or the end of the transcripts either matching as forward or reverse, Concordance Index and total number of observations (Table 1).

Note2: Since contigs with multiple matches aren't considered some hooks might be missing from the Step1-All_Hooks. This usually happens with low complexity hooks made of repetitive short sequences.

Hook	Start-Forward	Start-Reverse	End-Forward	End-Reverse	C.I.	Total Observations
a10-178	1	68	150	3	0.96	222
a11-1448	0	194	489	1	0.99	684
a14-50	0	15	49	9	0.75	73
a18-82	1	33	54	2	0.93	90
a1-9920	1476	4	5	3793	0.99	5278
a22-33	15	45	40	10	0.54	110
a27-96	0	26	54	0	1.00	80
a6-153	112	22	19	118	0.70	271
a9-68	0	0	3	0	1.00	3

Table 1: Example of a Step1-All_Hooks table.

- 2) Step1-Best_Candidates: Same as Step1-All_Hooks but only with the Best Hooks, plus the Concordance Index and total count number. The Median count value is included in the header (Table 2).

Hook	Start-Forward	Start-Reverse	End-Forward	End-Reverse	C.I.	Total Observations
a10-178	1	68	150	3	0.96	222
a11-1448	0	194	489	1	1.00	684
a1-9920	1476	4	5	3793	1.00	5278

Table 2: Example of a Step1-Best_Candidates.

- 3) Best-Hook.fasta: Sequences of the Best hooks

Additionally, two folders will be added inside Results:

- 1) Hooks: Sequence of all the Hooks and the CD-HIT-EST cluster report
- 2) Raw_Matches: Best Hooks matching sequences on the transcriptome.

A summary of all results is also provided on the file Step1_Summary located on the main Analysis Directory (the one provided with “-o” parameter). It includes the relevant parameter values used in the run, transcriptome files analyzed with the number of transcripts and the kmer counts filter applied to them, the tables from the files “Step1-All_Hooks table” and “Step1-Best_Candidates”, and the matches number found for each hook divided by transcript end and orientation.

SLFinder-step2

This script takes the Best Hooks raw matches sequences recovered on the previous step, filters them by size and sequence similarity, aligns them and then automatically trims the results to generate SL sequences. During its run, CD-HIT-EST is used in many steps to reduce sequence redundancy and shorten running times.

The initial filtering is carried out with “seqkit seq” to filter sort sequences and then cd-hit-est. They can be controlled with:

- c, --cdhitR: Sequence identity threshold used for Raw Matches filtration by clustering sequences with CD-HIT-EST. Default value is 1 (100% sequence identity).

-m, --minimun: Minimal Raw Match length to consider. Default value is 1 (No filtering)

Then the sequences are aligned using “mafft --globalpair”. While intended for a precision alignment of a small dataset, the relative short sequences (50pb maximum by default) allow to align several thousand sequences in minutes. Trimming is then carried out with Trimal and it can be controlled using:

-ts, --trimalsm: Similarity score used in Trimal. Default value is 0.001

-tg, --trimalgap: Gap threshold used in Trimal. Default value is 0.90

To ensure that the trimmed sequence properly represents the original region (i.e. no central position was removed, hindering SLFinder-step3 approach). The central section is retrieved using with “seqkit subseq -r 2:-2”, gaps removed and the sequence searched against the original alignment with “seqkit grep -s -r”. These bases are removed because more likely to be errors and terminal mismatches here aren’t critical on SLFinder-step3, unless “-sc” is set to 100 (see below).

If one of the sequences doesn’t match the automatic process for that file ends (we recommend manual curation of the alignment). Passing this control, gaps are removed and clustered with CD-HIT-EST again with a 100% identity threshold. First creating an individual file and then combining al sequences. In addition, a sequence logo is generated for the alignments pre and post trimming (Fig. 5) intended to aid the manual curation.

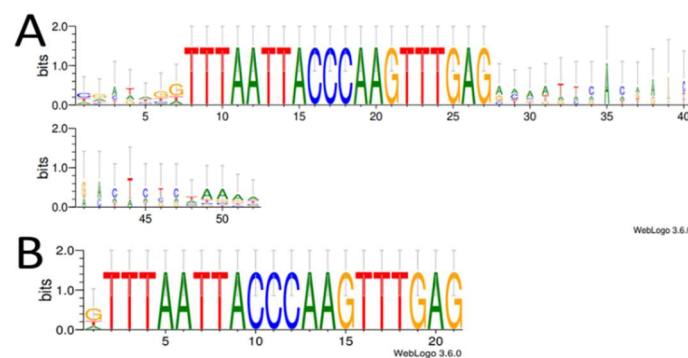


Fig. 5: Example of Sequence Logos for a hook matching SL-1 of *C. elegans* before (A) and after (B) the trimming process. Since the symbol “-” is not included in the Weblogos alphabet, positions with smaller letters are positions with more GAPS.

Other options to consider are:

-o, --outdir: Directory where to store the output files, must be the same as the string used on SLFinder-step1. The default name is “SL-analysis”

-t, --threads: Number of threads to use when parallelization is possible (CD-HIT-EST and MAFFT), default value is 1

A small overview of the results is provided in the file Step2_Trimal-Stats, which contains Basic stats of the trimmed sequences (seqkit stats). It includes the number of sequences, total length (irrelevant for this application), min, max and average sequence length (Table 3).

file	format	type	num_seqs	sum_len	min_len	avg_len	max_len
a1-4907-3prima-forward.trimal-nogap	FASTA	DNA	12	252	21	21	21
a1-4907-3prima-reverse.trimal-nogap	FASTA	DNA	1211	26486	15	21.9	22
a1-4907-5prima-forward.trimal-nogap	FASTA	DNA	143	3289	23	23	23
a1-4907-5prima-reverse.trimal-nogap	FASTA	DNA	2	62	31	31	31
a23-221-3prima-forward.trimal-nogap	FASTA	DNA	47	985	20	21	21
a23-221-5prima-reverse.trimal-nogap	FASTA	DNA	8	216	27	27	27
a8-907-3prima-forward.trimal-nogap	FASTA	DNA	46	1282	24	27.9	28
a8-907-5prima-reverse.trimal-nogap	FASTA	DNA	8	248	31	31	31

Table 3: Example of a Step2_Trimal-Stats file.

Additionally, two folders will be added to Results:

- 1) Hool_Variants: Trimmed sequences in Fasta format for each original alignment file (folder Hook_Individual), combined Hook_all_variants.fa and their respective cluster reports
- 2) Weblogos: Sequence logos for the raw and trimmed alignments (folders alignments and trimmed_alignments respectively).

A summary is provided on the file Step2_Summary located on the main Analysis Directory. It includes the relevant parameter values used in the run, hook variants generated and the subsets of sequences that were discarded.

SLFinder-step3

The final steps starts by making a BLAST search of all Hook variants (file: "Analysis directory"/Results/Hook_Variants/Hook_all_variants.fa by default) against the Reference Genome. Once identified, the script determines loci orientation by checking how many times its sequence is present, both in forward and reverse, on the raw sequences using "grep -c". Twenty percent of the loci length pb from both ends to account for incomplete sequencing. Since these are separated by transcript end and hook orientation, finding more hits in a file indicates orientation (only the hook sequences that generated the best Hook variant are checked). According to the results, the pipeline continues by searching for the donor site in the 3' end of the Loci ("GT" at the end of the Loci if it is encoded in Forward or "AC" at the beginning if in Reverse). If for some reason the Loci's sequence

is found an equal number of times present on Forward and Reverse (or it isn't found on the Raw sequences) the orientation is reported as "undetermined". Finally, the region surrounding the loci is annotated using a Protein/cDNA reference.

The script requires two inputs: A Reference Genome and an external reference with known proteins or cDNAs (e.g. Swiss-Prot from Uniprot). They can specify with these options:

- g, --genomeblast: Reference Genome. Must be an uncompressed Fasta file.
- br, --blast_ref: Protein or Reference database to be used to annotate the pSL genomic Loci. Must be an uncompressed Fasta file.

The pSL vs Reference Genome BLAST search is done with "blastn -task blastn-short", settled as ungapped and with a 100% identity threshold. Query coverage can be changed to account for non-matching pbs on the terminal regions (likely caused because sequencing errors or problems during the trimming process). A higher value reduces false positives, however if set at 100 some true SL sequences may not be detected due to problems during the automatic trimming.

- gc, --gen_cov: Coverage threshold of the pSL hits on the genome. Default value is 90 (up to two not matching pbs on 20 pb sequence either end in a 20pb Hook Variant).

Hits are then sorted by starting coordinate and the individual pSL each Loci identify by checking which pSL hits overlap and if there are anomalies. If the pSL variants are in fact SL sequences they should overlap almost on their entirety. The pipeline check this by taking the first hit according to its coordinates and checking if the following matches overlap with it. If it does and the End coordinate is further away the loci coordinates are extended. Depending on the how big is the last loci compared with the longest hit the scripts registers an alert number: 0 (no problem) if its extended less than 2pb, 1 (acceptor site determination might not reliable for some of the pSL variants) if its extended between 2 and five pb, 2 (acceptor site determination is not reliable and it is not analyzed) if between 2 and 5 pb and 3 (probably a microsatellite) if more than five. These configurations are illustrated on Fig 6.



Fig. 6: Possible ungapped alignment configuration between pSL variants and their coding loci. A represents the ideal situation where all pSL matching the loci overlap

almost entirely while (Alert 0) C is likely false positive in a low complexity region (Alert 3). Configuration B (Alerts 1 and 2) are given where some pSL extend slightly more than the others is more likely to be caused by retention of some pb from the transcripts that also match with the genome.

As mentioned above, Donor sites are identified by simply counting either “GT” or “AC” in the 3’ end of the pSL according to its orientation. Testing however shows that is common the inclusion of a limited number of base pairs at both the 5’ and 3’ ends of the Hook Variants that can then be incorporated in the reported pSLs when they match the Genomic sequence. These often are “G” or a “GT” matching the real Donor site but longer matches has been observed.

Because of this, the script first scan 4 pb surrounding the pSL 3’ coordinate to detect the donor site, and then verifying if it is or isn’t included in the hook variants. Also, Loci with Alerts 2 or 3 are not analyzed and require manual inspection. The information needed will be stored in the folder “Genome_locis” and divided by chromosome (or contig/scaffold name): A table with all pSL variants matching the loci and an alignment produced with MAFFT (Table 4 and Fig 7). Another situation that can happen is that only some of the Hook Variants matching the loci have a potential donor site in the expected region. If more than 20% have this issue the loci is reported as “Unclear” and it will be treated separately (see below).

Hook Variant	Length	Start	End	Hook Orientation	Potential Donor site	Curation needed	Loci Orientation: Reverse
a10-178-3prima-forward-7	24	7140530	7140553	plus	1	yes	
a10-178-3prima-forward-60	23	7140531	7140553	plus	1	yes	
a10-178-3prima-forward-61	22	7140532	7140553	plus	1	no	
a10-178-5prima-reverse-3	23	7140532	7140554	minus	1	no	
a10-178-5prima-reverse-9	23	7140532	7140554	minus	1	no	

Table 4: Example of an individual loci table detailing each matching pSL, its length, coordinates in the chromosome, orientation, potential donor sites and if curation is advised: there was no potential donor site or the Hook Variant overlaps with it. Since the program takes four bases surrounding the 3’ region coordinates these values can be 0, 1 or 2. The loci orientation is included on the Header.

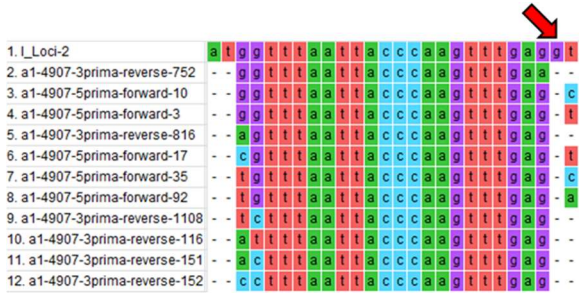


Fig 7: Alignment between the Loci and each matching Hook variant, obtained with MAFFT and displayed on MEGA7. The arrow signals the potential donor site.

The next step is a BLAST search against the external reference for annotation. Since most false positives with a biological origin at this stage are likely 5' UTR sequences a cDNA reference is preferable, but often this is not available. Also, if one of the cDNAs includes the SL sequences by error the SL will be wrongly discarded. This search can be customized with the following parameters:

-bt, --blast_ty: Reference database type: "nucl" or "prot" for nucleotides and protein, depending on the selection, the script runs blastn or blastx. By default is "prot".

-be, --blast_ev: Order of magnitude for the evalue cutoff used in the BLAST annotation, the program reads it as '1e-[YOUR VALUE]'. Default value is 10, for a 1e-10 cutoff.

-gr, --rangeG: Region surrounding the pSL hit to annotate. Default value is 100.

For a pSL to be considered validated three points need to be met

- 1) There must be a matching locus in the Reference Genome
- 2) A potential donor site must be present
- 3) No hit must be found in the annotation BLAST.

Sequences are retrieved according to their orientation (Leaving the donor side "GT"), clustered with CD-HIT-EST (100% identity threshold) and reported in "SL.fa". The same is done with Loci with an "Unclear" donor site and are reported in the file "Unclear_SL.fa". In both files each pSL header will include the Loci number and the hook ID that identified it and had an identifiable donor site (referred as "Best SL ID" in Step3_Loci.tab. Lastly, the transcripts IDs with those hooks are retrieved from the temp_files folder, counted and organized in the folder "Spliced_leader_transcripts". One important note: these are all transcripts with hits for the original hook, not the ones bearing the pSL variant found in the genome. They are provided as a quick overview of how well supported is the pSL and facilitate to discard false positives as Histone genes. See "Working with the pSL sequences for details".

Other options to consider are:

-o, --outdir: Directory where to store the output files, must be the same as the string used on SLFinder-step1. The default name is "SL-analysis".

-s, --pslfile: File name with trimming information needed to retrieve the pSL sequences (Default: "Analysis directory"/Results/Hook_variants/Hook_all_variants.fa)

-t, --threads: Number of threads to use when parallelization is possible (BLAST, CD-HIT-EST and MAFFT), default value is 1

Main results are provided three files:

- 1) Step3_Loci.tab: Table summarizing the SL Locus. It includes their location, longest and best SL hits, donor sites present and the annotation Blast search results (Table 5).
- 2) SL_final.fa: Fasta file with the validated SLs.
- 3) Step3_Spliced_transcripts: Matching transcript counts for each hook with at least one “Best pSL ID” in a Loci, across all analyzed transcripts (Table 6).

Additionally, two folders will be added inside Results:

- 1) SL_BLAST: Results of the SL Blast search against the reference genome
- 2) Genome_locis: Each loci hits summary (table with matching pSL variants to the loci and alignment file) separated in folder per chromosome, and results of the annotation BLAST (Blast_annotation folder).
- 3) Spliced_leader_transcripts: Folder with the Transcripts IDs of the Hooks that identified the SLs on each assembly, separated on different folders.

A summary is also provided on the file Step3_Summary located on the main Analysis Directory. It includes the relevant parameter values used in the run, the total number of pSL Loci detected and donor site detection count. It also includes tables for the files Step3_Spliced_transcripts and Step3_Loci.tab, however Step3_Loci.tab is subdivided according to the donor site detection and the clusters generated by CD-HIT-EST to identify representative pSL sequences. First are Loci with a clear donor site and each Cluster (with the Loci representative of the Loci and it's sequence), the Unclear Loci and their clusters, and the discarded Loci (Loci close to a CDS sequence, Donor site not detected and Not analyzed).

Warning files

Depending on your data and the references used several problems can arise at different steps of the pipeline. Reports of these are saved on the Warnings folder.

- 1) Step1-Hook_Anomalies: Results of the self-BLAST search among the Hooks. It only contains Query Hook, Subject Hook and alignment Start and End positions in the query (Table 7).
- 2) Step2_Manual-curation: Report of the files that require to be manually curated (Trimal brought no results, sequence was too short, or the center was modified)

N Loci	Scaffold	Loci extended length	Loci Orientation	Hook Variants	Donor Site	Best pSL ID	Start-Best	End-Best	Annotation Hits	Best Hit ID	Identity	Evalue
Loci-1	I	21	Forward	10	3prima*	a7-135-3prima-reverse-8	4172208	4172228	15	O61790	100	6.97E-51
Loci-2	I	22	Forward	13	3prima	a1-4569-5prima-forward-3	8781697	8781718	0	-	-	-
Loci-3	I	21	Forward	8	3prima*	a1-4569-5prima-forward-48	11709195	11709215	0	-	-	-
Loci-4	II	14	Undetermined	1	Not Analyzed	a1-4569-3prima-reverse-806	3032461	3032474	17	A8I499	44.304	5.05E-13
Loci-5	II	21	Forward	10	3prima*	a7-135-3prima-reverse-8	5543600	5543620	0	-	-	-
Loci-6	II	21	Reverse	10	5prima*	a7-135-3prima-reverse-8	5544251	5544271	6	P70584	68.421	6.20E-17
Loci-7	II	21	Forward	10	3prima*	a7-135-3prima-reverse-8	5844867	5844887	19	Q9TVW5	71.311	3.02E-48
Loci-8	V	28	Reverse	26	Unclear	a1-4569-5prima-forward-3	17121745	17121766	0	-	-	-
Loci-9	V	28	Reverse	26	Unclear	a1-4569-5prima-forward-3	17122725	17122746	0	-	-	-
Loci-10	V	28	Reverse	26	Unclear	a1-4569-5prima-forward-3	17123701	17123722	0	-	-	-
Loci-11	V	28	Reverse	26	Unclear	a1-4569-5prima-forward-3	17124673	17124694	0	-	-	-
Loci-12	V	28	Reverse	26	Unclear	a1-4569-5prima-forward-3	17125649	17125670	0	-	-	-

Table 5: example of a Step3_Loci.tab table. It summarizes the Loci location, length, matching SL, donor site and annotation results. Best pSL ID is the longest matching Hook variant with an identified donor site if present, the longest one if doesn't. The asterisk "*" in some Loci reported either 3 or 5prima Donor indicates that the best Hook variant overlaps with the potential donor site.

Hooks	SRR5832182	SRR5832183	SRR5832184	SRR5832185	SRR5832186	SRR5832187	SRR5832188	SRR5832189
a1-4569	376	363	307	395	386	354	323	425
a7-135	14	18	10	20	26	13	8	18

Table 6: example of Step3_Spliced_transcripts file.

Query-Hook	Subject-Hook	Alignment Length	Start	End
a1-4907	a1-4907	32	1	32
a1-4907	a8-907	14	21	8
a8-907	a8-907	21	1	21
a8-907	a1-4907	14	26	13

Table 7: Example of a Step1-Hook_Anomalies. Notice that includes the line of each hook with itself, this redundancy helps when the hook has a repetitive sequence and matches with itself multiple times.

- 3) Step3_Sequence-retrieval-warnings: Report with the Loci were not enough sequence could be recovered for the annotation BLAST search or even the Donor site determination. This happens if the pSL is located too close to the end of a chromosome/coting in the reference genome.
- 4) Step3_Problematic-Loci: Reports loci whose apparent length when recovering the genomic loci is larger than the largest individual hit. This can happen in low complexity regions.

Working with the pSL sequences

If there are no surprises with your particular data and the sear was fruitful, you should have a manageable number of potential pSL. Next it is needed to discard false positives as Histone genes or adapters used during sequencing that for some reason made their way into the transcriptome assembly. Retrieving the cotings matching each hook with an pSL and annotating them is probably the most straight forward approach. After identifying and discarding those we recommend use another pipeline specialized on identifying the trans-spliced genes themselves given the known sequence of the SLs, like SL-Quant (<https://github.com/cyaguesa/SL-quant>).

Remember that our pipeline is prone to report several artificial variants of the same SL by including extra pbs at the sequence ends. Particularly “G” or “GT” in the 3’ end (that aligns with the donor site in the genome). So manual trimming of the pSLs might be required. For the same reason do not discard Loci with “Unclear”, “Not detected” or “Not Analyzed” donor sites without verifying the alignments stored on Genome_locis”

How to...

- 1) Use a transcriptome assembled with a program other than Trinity or filtered following a custom strategy.

As stated before, this can be done by properly using “-a” and “-f” in Step1. As long there is one sequence for every true gen (or close to) the pipeline should work without issue.

The only thing important to consider is the transcripts headers. Make sure that each one has a distinct identifier and that they do not include special characters in regular expressions like points. Formats like Ensembl (e.g. ENSMUST00000021802.15) are ok because the point is always the character N°19, but if the position isn't constant across transcripts Step1 is likely to fail. If your assembler does this replace the point for “_” or other non-special character before running the pipeline.

2) Filter hooks that are known to not be SL

If too many Best Hooks are generated running SLFinder-step1 it can save time to make a BLAST search against a cDNA reference to identify false positives (e.g. Histones) and/or remove anomalous hooks reported on “Step1-Hook_Anomalies” file. To do this just remove the hook sequence files located on: “Analysis directory”/Results/Raw_Matches

3) Use manually curated alignments on SLFinder-step 3

If needed, you can find the unaligned sequences on “\$OUT_DIR/Results/Raw_Matches” and the alignments on “Analysis directory”/Results/pSL_variants/pSL_Individual, checking the sequence logos for the alignment might be also a good starting point located on “Analysis directory”/Results/weblogos/alignments to diagnose the problem. Once curated, the alignment can either be trimmed with Trimal and processed:

```
trimal -gt [0.90] -st [0.001] -fasta -in [input_file] | awk '/^>/ {printf("\n%s\n", $0); next; } { printf("%s", $0); } END {printf("\n");}' | awk 'NF' | grep -v ">" | awk -v data=[Identifier] '{print ">data-" NR "\n" $s}' >> output_file
```

Red parameters need to be changed according to your file, blue ones are the default values used by SLFinder-step2. The rest of the command line linearizes the fasta sequence and renames each sequence as “Identifier”-Number. In the script this identifier is the one used for the files ([HookID]-[contig end]-[match orientation]) but there is no need to maintain it.

If you decide to also trim the alignment manually just run this part of the command for the reformat and renaming:

```
awk '/^>/ {printf("\n%s\n", $0); next; } { printf("%s", $0); } END {printf("\n");}' | awk 'NF' | grep -v ">" | awk -v data=[Identifier] '{print ">data-" NR "\n" $s}' >> [output_file]
```

Once completed, concatenate all sequence files and remove gaps running:

```
cat [your trimmed files] | seqkit seq -g >> [concatenated_file]
```

And finally, cluster sequences by sequence similarity:

```
cd-hit-est -c 1 -i [concatenated_file] -o [pSL_file]
```

Use the “-s” parameter in SLFinder-step3 to use this new pSL_file.

It's important that you do not change the name if the Hook Variants. This is used in Step-3 to verify Loci Orientation.