GeneMiner Manual

1. Overview
2. Download and install
- 2.1 GeneMiner with Graphical User Interface (GUI)
- 2.2 **GeneMiner with command (cmd)**
  - 2.2.1 Cloning the GitHub repository (support)
  - 2.2.2 Source code installation
  - 2.2.3 Flexible construction

3. Quick start
- 3.1 Running GeneMiner-cmd
- 3.2 Running GeneMiner-GUI

4. Detailed User Guide
- 4.1 Input
  - 4.1.1 Reads Files
- 4.1.2 Reference files
- 4.2 Explanation of parameters
  - 4.2.1 Basic parameters
  - 4.2.2 Advanced parameters
- 4.3 Output
  - 4.3.1 reference_database
  - 4.3.2 filtered_out
  - 4.3.3 assembled_out
  - 4.3.4 GM_results
  - 4.3.5 bootstrap_out
  - 4.3.6 results.csv

5. Example
- 5.1 Mining chloroplast genes from genome skimming data
- 5.2 Mining mitochondrial genes from genome skimming
- 5.3 Mining nuclear genes from genome skimming
- 5.4 Mining Angiosperms353 genes from transcriptome data

6. Methods
- 6.1 Filtering reads
- 6.2 Assembly
- 6.3 Verifying results

7. Contact
8. Citation

1. Overview

GeneMiner is a software for extracting phylogenetic markers from next-generation sequencing (NGS) data. (i) With GeneMiner, users can accurately and efficiently obtain a large number of phylogenetic markers from NGS data at an economical cost. For example, extract all or part of mitochondrial/chloroplast genes and highly repetitive regions (e.g., nrDNA) in the nuclear genome from genome skimming data. Extract single-to-low-copy genes from transcriptome sequencing data, etc. GeneMiner broadens the choice of phylogenetic markers from the most basic data level. (ii) GeneMiner proposes a novel verification method based on the base substitution model and repetitive resampling, which can statistically evaluate the impact of the reference on the assembly. (iii) GeneMiner provides a cross-platform graphical interface and can be easily docked

to downstream phylogenetic analysis processes. In addition, GeneMiner can be applied to the research of DNA barcode extraction, customs quarantine, specific functional gene exploration, and other research, which has broad application prospects.

2. Download and install

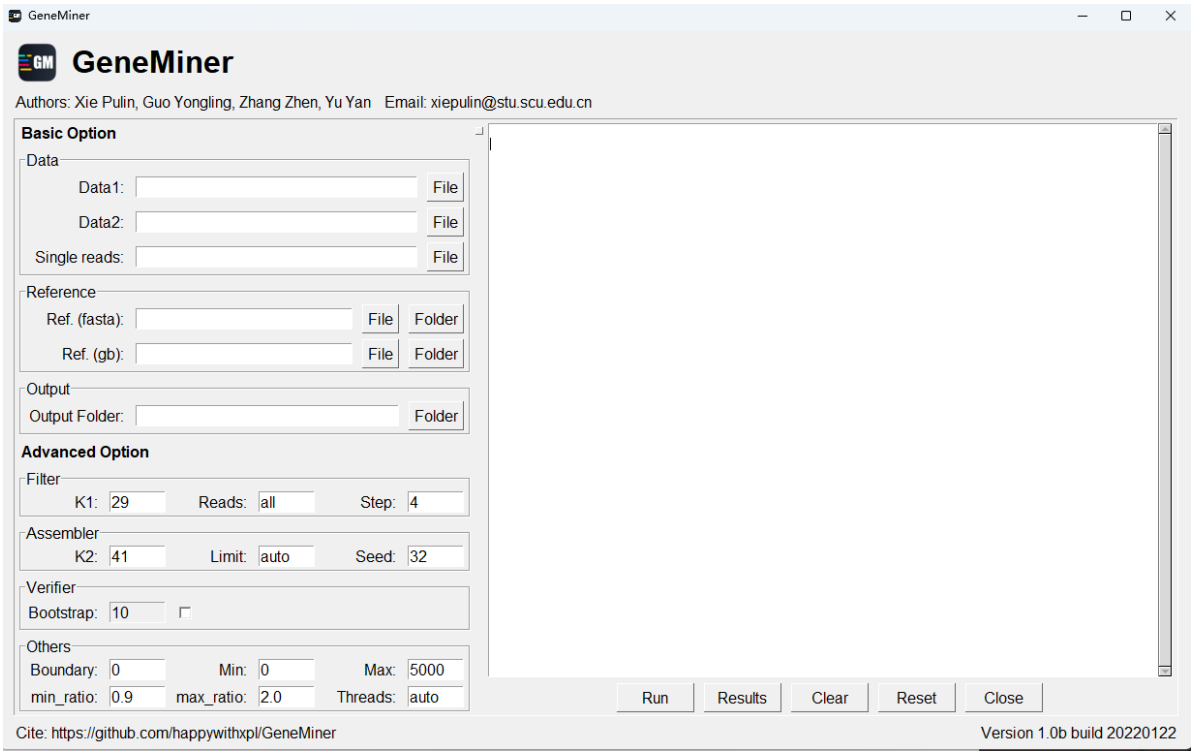
GeneMiner is open source under the GPL-3.0 license, distributed through the GitHub repository (https://github.com/happywithxpl/GeneMiner/releases). Please follow our GitHub page to stay up-to-date with the latest code changes. We do not provide any support for previous versions of the code! Version numbers follow the notation x.y.z, where x changes with major code reorganizations, y changes when new features are added, and z changes with bug fixes.

GeneMiner is an easy-to-use software written in python3, which is provided for x86-64 systems running GNU/Linux, macOS (version 10.13 or higher), and Windows (64-bit, version 7 or higher).

Windows, macOS, and Linux users can run GeneMiner directly from the command line. We also offer a more convenient GUI version for Windows and macOS users.

2.1 GeneMiner with Graphical User Interface (GUI)

We strongly recommend using the GUI version for users who are not familiar with the command line or light use.Download the corresponding version of the packaged GUI from https://github.com/happywithxpl/GeneMiner/releases and double click to run it.



2.2 GeneMiner with command (cmd)

2.2.1 Cloning the GitHub repository (support)

Clone GeneMiner's repository directly and build it as below:

```
git clone https://github.com/happywithxpl/GeneMiner.git
cd GeneMiner
python setup.py install --record logName --user #Add 'geneminer.py' to the
'$PATH'
geneminer.py -h
```

If you want to remove `geneminer.py` from the `$PATH`, you can:

```
cat logName | xargs rm -rf
```

2.2.2 Source code installation

Download the source distribution from the release and install dependencies:

```
wget -c
https://github.com/happywithxpl/GeneMiner/releases/download/v1.0.0/GeneMiner_v1.
0.0_linux.tar.gz
tar GeneMiner_v1.0.1_linux.tar.gz
cd  GeneMiner_v1.0.1_linux
python setup.py install --record logName --user
geneminer.py -h
```

If you want to remove `geneminer.py` from the `$PATH`, you can:

```
cat logName | xargs rm -rf
```

2.2.3 Flexible construction

If both of the above methods fail or you want to have a deeper control of GeneMiner, you can use a more flexible method.

- Download the GeneMiner's distribution from here.

```
wget -c
https://github.com/happywithxpl/GeneMiner/releases/download/v1.0.0/GeneMiner_v1.
0.0_linux.tar.gz
tar GeneMiner_v1.0.1_linux.tar.gz
```

- Use the following commands to make `geneminer.py` executable.

```
cd GeneMiner_v1.0.1_linux
chmod 755 geneminer.py
```

- Install python library `biopython` using pip or conda.

```
# install required libs
pip install biopython --user
```

- Add `geneminer.py` to the `$PATH`. The following is an example for linux users:

```
echo "export PATH=\$PATH:$(pwd)" >> ~/.bashrc
source ~/.bashrc
geneminer.py -h
```

3. Quick start

3.1 Running GeneMiner-cmd

Before using GeneMiner to mine particular target genes, we strongly advise you to be aware of the state of your sequencing data (from Illumina, Roche-454, ABI, or other sequencing platforms), including the method, depth, quality, data size, etc. This facilitates the choice of parameters.

GeneMiner takes the reference and fastq format sequencing files as input and the recovered phylogenetic markers as output. We have prepared a simulated dataset of `Arabidopsis thaliana` to help you quickly grasp the primary usage of GeneMiner. You can download them from https://github.com/happywithxpl/GeneMiner-Test/tree/main/Demo

(1) Mining single target gene

```
geneminer.py -1 skimming_data1.fq.gz -2 skimming_data2.fq.gz -rtfa
shallow_ref/ITS.fasta -o ITS_out
```

`-1`,`-2`: the input files with paired-end reads, given in FASTQ format

`-rtfa`: reference sequences in FASTA format

`-o`: the output directory

NOTE: The reference sequences are derived from homologous genes of closely related species (same genus or family). If a reference sequence has multiple homologous genes from one or more species, please store these homologous genes in a FASTA format file as a reference file .The file may look like this:

```
>species1_GeneA
ATCGATCG
>species2_GeneA
ATCGATCC
>species3_GeneA
ATTGATCC
>species4_GeneA
ATTGATCT
```

(2) Mining multiple target genes

GeneMiner allows multiple reference files to be stored in the same folder to facilitate batch mining of target genes.

```
#for FASTA format
geneminer.py -1 skimming_data1.fq.gz -2 skimming_data2.fq.gz -rtfa shallow_ref/
-o out1
#for GenBank format
geneminer.py -1 skimming_data1.fq.gz -2 skimming_data2.fq.gz -rtgb
shallow_ref.gb -o out2
```

`-rtfa`: reference sequences in FASTA format

`-rtgb`: reference sequences in GenBank format

(3) Accuracy assessment

```
geneminer.py -1 data1.fq.gz -2 data2.fq.gz  -rtfa  ref.fasta -bn 100 -o out
```

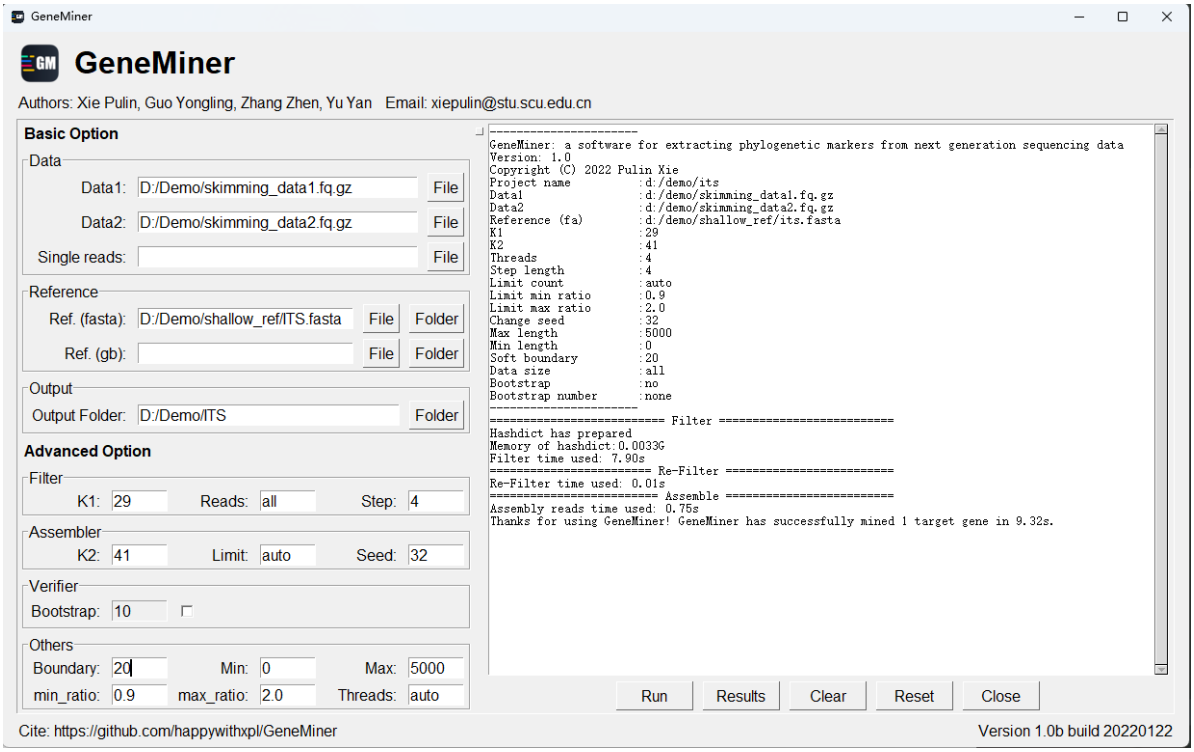
`-bn`: Specify the bootstrap number. Evaluate the assembly results based on repeated resampling and base substitution model.

Based on the base substitution model and repetitive sampling, GeneMiner creatively suggests a verification approach that can statistically assess the influence of the reference on the assembly results and confirm the stability of the assembly results.

3.2 Running GeneMiner-GUI

GeneMiner-GUI is straightforward, simple to use, and ideal for lightweight users. Considering the memory limit and the excessive time overhead, we recommend you utilize GeneMiner-cmd when you run large-scale data

For GeneMiner-GUI, you must set `Data1`, `Data2` or `Single reads` under the `Data` module, `Ref.(fasta)` or `Ref.(gb)` under the `Reference` module and `Output Folder` under the `Outout` module as below:



Run
Set the parameters, then click on the `Run` botton

View Results
Click on the `Results` button to view the results after the GeneMiner has completed

Clear the screen
Click on the `Clear` botton

Reset parameters
Click on the `Reset` botton and all parameters will be reset to default parameters.
WorkFlow: Reset ==>> Enter new parameters ==>> Run

Note: Do not click on the `Reset` botton while the program is running

Close GeneMiner
Click on the `Close` botton
Note: If closing GeneMiner's GUI does not kill the scripts in the background, you can open Task Manager to close GeneMiner's tasks.

4. Detailed User Guide

4.1 Input

4.1.1 Reads Files

GeneMiner assembles any next-generation sequencing (NGS) reads given in the FASTQ format. Giving paired or unpaired reads as input are OK and keep in mind that GeneMiner will use pairing information. GeneMiner can directly read compressed files in `gzip` . Compressed files should end with `.fq.gz` or .`fastq.gz`.

```
1   @ST-E00600:235:HFCTTALXX:1:1101:4752:1016 1:N:0:GGAGAACA
2
NCCCTATCATTTCTGAGGGGTTACATTCTCATTCTCTCAACATAAATACAGAGACCACTTCACCAAACACACCAACTCTG
TCTCTGGATGTTGATATGATAAACACATCAATTCCTGACACTCC
    ATCTTAGTTCTTGGAGAGGCACCACT
3   +
4   #-FFFJJJJJJJJJJJ7JJ7J7JJJJ-J--JJ-JJJ-JJ7JJJJJJJJ-JJJ7J--J-J-JJJ-JJJ--JJJ--J--
JJ-JJJJ7--J7--J7JJJJJJ77JJJJ7J-7JJ---J-JJ--JJ7-     J7JJ---JJ-JJ-JJJJJJJ-F>F>F
```

4.1.2 Reference files

DNA sequences are stored in reference files in FASTA format or GenBank format. Each reference file contains one or more DNA sequence(s), which is the homologous gene from closely related species (same genus or family). In addition, GeneMiner allows multiple reference files to be stored in the same folder to facilitate batch mining of target genes.

(1)For a reference in FASTA format, it may look like this:

```
>Aegopodium_podagraria
TCGAATCCTGTGATAGCAGAACGACCCGCTAACTGGTAAATATATTGGGCAAGCTCATGGGGATTTTATCCCCTGTTGGT
GAACCCTTGGTAGGTGGTCACTCCCCGGTTGCCACTGGCC
>Anethum_foeniculum
TCCTCCTTATCATTAGAGGAAGGAGAAGTCGTAACAAGGTTTCCGTAGGTGAACCTGCGG
>Bupleurum_chinense
AACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTCGAATCCTGAATCGAAGAGCGACCCGAGAACATGTTTTAA
GACGGGGCCAGCGGTCGTCGGCCTCGGCCTGTCGGCTGCG
>Chamaesium_paradoxum
AAGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTCGATGCCTGCAACAGCAGAATGACCCGTGAACACGT
ATAAAACATTGGGCTAGTAGATGGGGCGCAAGTTCCCGGA
CATGAACCCCAGGACGGATG
```

NOTE: The reference's `filename` will be used as the name of the target gene in GeneMiner.

(2) For a reference in GenBank format, it may look like this:

```
LOCUS       Daucus_carota         155895 bp    DNA     circular UNA 28-FEB-2021
DEFINITION  Daucus carota chloroplast, complete genome.
ACCESSION   urn.local...2i-d2j4m7m
VERSION     urn.local...2i-d2j4m7m
KEYWORDS    .
SOURCE      Daucus carota (carrot)
  ORGANISM  Daucus carota
            Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
            Spermatophyta; Magnoliophyta; eudicotyledons; Gunneridae;
            Pentapetalae; asterids; campanulids; Apiales; Apiaceae; Apioideae;
            Scandiceae; Daucinae; Daucus; Daucus sect. Daucus.
FEATURES             Location/Qualifiers
     source          1..155895
                     /organism="Daucus carota"
                     /organelle="plastid:chloroplast"
                     /mol_type="genomic DNA"
     tRNA            complement(6..77)
                     /gene="trnH-GUG"
                     /product="tRNA-His"
                     /note="anticodon:GUG"
                     /standard_name="trnH-GUG tRNA"
     gene            complement(6..77)
                     /gene="trnH-GUG"
                     /standard_name="trnH-GUG gene"
      ...

    ORIGIN
        1 ttgggcgaac gacgggaatt gaacccgcgc gtggtggatt cacaatccac tgccttgatc
       61 cacttggcta catccgcccc gccagttttc ttttatttat ttgcatttca aaggattcct
      121 ttttgatcat tcaaaaatat ttgtttatct aaaaaagtct taaataaata aaaaaggagc
      181 aagaccgcct cttgatagaa caagaaagag gttattgctc ctttttaat atttcaaaaa
      ...
   155821 cgttcactaa aaaaaaaccc ttttgtagca aatcgtttat taagaaaaat tgataacctt
   155881 aacacaaaag cagaa
   //
```

NOTE: The value corresponding to the Gene in `FEATURES` will be used as the name of the target gene.

4.2 Explanation of parameters

Using `geneminer.py -h`, the computer will display all options.

```
usage: geneminer.py <-1 -2|-s>  <-rtfa|-rtgb>  <-o>  [options]

GeneMiner: a tool for extracting phylogenetic markers from next-generation
sequencing data
Version: 1.0.1
Copyright (C) 2022 Pulin Xie
Please contact <xiepulin@stu.edu.scu.cn> if you have any questions

optional arguments:
  -h, --help          show this help message and exit

Basic option:
  -1                  File with forward paired-end reads (*.fq/*.fq.gz)
  -2                  File with reverse paired-end reads (*.fq/*.fq.gz)
  -s , --single       File with unpaired reads (*.fq/*.fq.gz)
  -o , --out          Output folder
  -rtfa <file|dir>    Reference of the target gene(s) only supports FASTA format
  -rtgb <file|dir>    Reference of the target gene(s) only supports GenBank
format

Advanced option:
  -k1 , --kmer1       Length of kmer for filtering reads [default = 29]
  -k2 , --kmer2       Length of kmer for assembling reads [default = 41]
  -d , --data         Specify the number of actually used reads to reduce the
computational burden (e.g.                          10000000)
                      Set to all if you want to use all the data [default = all]
  -step_length        Length of the interval when splitting the reads into k-
mers [default = 4]
  -limit_count        The minimum number of times a k-mer should appear in
reads,
                      used to remove likely erroneous and low-abundance k-mers
[default = auto]
  -limit_min_ratio    Minimum ratio of the recovered target gene(s) to the
reference's average length                          [default = 0.9]
  -limit_max_ratio    Maximum ratio of the recovered target gene(s) to the
reference's average length                          [default = 2.0]
  -change_seed        Times of changing seed [default = 32]
  -scaffold           Make scaffolds (in beta)
  -max                The maximum length of contigs to be retained [default =
5000]
  -min                The minimum length of contigs to be retained [default = 0]
  -t , --thread       Number of threads [default = auto]
  -b , --boundary     Length of the extension along both sides of the recovered
target gene
                      Set to a large value (e.g. 10000) if you want to retain
the complete assembly
```

```
                      Recommended length is 0.5 * reads length [default = 75]
  -bn , --bootstrap   Specify the bootstrap number [default = 50]
                      Evaluate the assembly results based on the base
substitution model and repeated                                resampling
```

4.2.1 Basic parameters

`-1` and `-2` : The input files with paired-end reads, given in FASTQ format (*.fq*.fastq*.fq.gz*.fastq.gz). Make sure to keep the correct file extensions

`-s` : The input file with unpaired reads given in FASTQ format (*.fq*.fastq*.fq.gz*.fastq.gz). Make sure to keep the correct file extensions.

`-o , --out` : The output folder.

`-rtfa`: <file|dir> : The reference of the target gene(s), only support FASTA format. (*.fasta*.fa*.fas*.fna). Make sure to keep the correct file extensions

NOTE:

(1) The reference sequences are derived from homologous genes of closely related species (same genus or family). If a reference sequence has multiple homologous genes from one or more species, please store these homologous genes in a FASTA format file as a reference file. The file may look like this:

```
>Gene_A species1
ATCGATCG
>Gene_A species2
ATCGATCC
>Gene_A species3
ATTGATCC
```

(2) The memory consumption of the reference hash table is linearly related to the size of the reference sequences, and extensive reference sequences will dramatically increase memory consumption and significantly reduce the filtering speed. Based on the length characteristics of the phylogenetic markers, we suggest that each reference sequence length is 300-5000 bp, and the total entries do not exceed 100000.

`-rtgb` <file|dir>: The Reference of target genes, only support GenBank format (*.gb). Make sure to keep the correct file extensions.

NOTE:

GeneMiner does not assemble the entire plastid genome, but only mines the `gene` (feature=="gene") of the GenBank file.

Users can also edit the genbank file to specify specific target genes.

4.2.2 Advanced parameters

`-k1, --kmer1`: Length of k-mer for filtering reads. This parameter is used to filter the reads that are highly matched with the reference. We split the reference sequences into k-mers and stored them as reference hash tables. Each read in the original data is subsequently split into (`L`-`k1`+1) k-mers, where `L` is the read length and `k1` is of the same size as the setting of the reference. If a subsequence of a read appears in the reference hash table, the read is retained and assigned to the target gene's filtered dataset with labels. The accepted value of `k1` is related to the relationship of the species corresponding to the reference and sequencing data. The closer the relatives, the greater the tolerance of `k1`. Conversely, the more distant the relatives, the smaller the accepted value of `k1`. Generally speaking, a larger `k1` is favorable to improve the accuracy of the assembly. `k1` takes a range of values from 17 to 127 bp,default=29

`-k2, --kmer2`: Length of k-mer for assembling reads. `k2` is the length of nodes in the de Bruijn Graph, and GeneMiner assembles k-mers into contigs based on the overlap of k-1 mer between nodes. In general, smaller k-mers facilitate sequence extension at lower coverage but may introduce erroneous k-mers; larger k-mers handle reverse repetition efficiently but may terminate sequence extension early. For Illumina reads (150bp) with sufficient coverage (> 20x), we have good results with k = 41. `k2` takes a range of values from 17 to 127 bp,default=41

`-d, --data`: Specify the number of actually used reads to reduce the computational burden (e.g. 10000000). GeneMiner allows mining target genes, especially those with medium-high copy numbers, using a portion of the raw sequencing data. If you need to use the entire raw sequencing data as input, you can set `-d` as `all`. default = `all`

`-step_length`: Length of the interval when splitting the reads into k-mers. With a sequence is ATCGAATTCA, when `step_length` is 1, and `kmer_size` is 5, we can get ATCGA, TCGAA, CGAAT, GAATT, AATTC, and ATTCA; when `step_length` is 2, we will get ATCGA, GAATT, AATTCA. This parameter can be used to reduce the program runtime when the dataset is large with sufficient coverage. We utilized the `step_length` parameter to accelerate read filtering without dramatically sacrificing accuracy.

`-limit_count`: he minimum number of times a k-mer should appear in reads. This parameter is used to remove erroneous, low-quality k-mers. During the assembly process, GeneMiner will split the filtered reads into subsequences (k-mers) of length `k1` and count the number of occurrences of these k-mers. The k-mers whose frequency is below the `limit_count` will be removed. For example, a `limit_count` has a value of 3, which means only the k-mer with at least 4 occurrences will be used for assembly.

Users can either set a uniform `limit_count` or choose the `auto` mode. In `auto` mode, GeneMiner will assign a reasonable `limit_count` to each target gene based on the k-mers frequency distribution. default = `auto`

`-limit_min_ratio`: Minimum ratio of the recovered target gene(s) to the reference's average length. During the assembly process, GeneMiner discards contigs with ratios smaller than `limit_min_ratio`

`-limit_max_ratio`: Maximum ratio of the recovered target gene(s) to the reference's average length. During the assembly process, GeneMiner discards contigs with ratios larger than `limit_max_ratio`.

NOTE: GeneMiner applies some additional processing to the average length of the reference sequences. GeneMiner first sorts the reference sequences by length and then eliminates the short reference sequences. Finally, the average of the remaining reference sequences is taken and used as the average length that GeneMiner really works with.

`-change_seed`:Times of changing seeds. GeneMiner automatically replaces candidate seeds to achieve optimal assembly. default=32

`-max` : The maximum length of contigs to be retained. default = 5000 bp

`-min`: The minimum length of contigs to be retained. default = 0 bp

`-t, --thread`: Specify the number of threads. If not specified, GeneMiner will automatically select the appropriate number of threads based on computer performance. default = `auto`. In practical tests, balancing computational resources and speed, we recommend using 4 threads to get a superior overall performance.

`-b, --boundary`: Length of the extension along both sides of the target gene (length of `soft boundary`). When extending along both sides of the recovered target gene, the accuracy of the bases gradually decreases as the extension length increases. However, the decrease in accuracy is not precipitous but a gradual decline within a buffer of some certain length. We will keep the buffer and call this buffer a `soft boundary`. The recommended length is 0.5 * reads length. default =75 bp

Note:

(1) High quality close reference sequences and high target gene sequencing depth, `-b` recommended 0 to 150.

(2) Ordinary quality reference sequence and shallow target gene sequencing depth ，`-b` recommended > 300

(3) If you want to keep the recovered target genes for as long as possible, you can set `-b` to a very large number, e.g. 10000. In fact, both sides just retain as much base information as possible and cannot reach the specified large number (e.g. 10000)

`-bn, --bootstrap`: Number of resampling. GeneMiner borrowed ideas from Bootstrap, and then innovatively proposed a method based on the base substitution model and repeated resampling, which can effectively evaluate the accuracy of the results. default = 50

4.3 Output

The output directory contains the `reference_database`, `filtered_out`, `assembled_out`, `GM_results`, `bootstrap_out` and `results.csv`

4.3.1 reference_database

`reference_database` <folder>:

Used to store reference sequence datasets. GeneMiner preprocesses the reference sequences of the close taxa provided by the user and writes these reference sequences to FASTA format files.

4.3.2 filtered_out

`filtered_out` <folder>:

Used to store filtered datasets. GeneMiner searches the raw sequencing data based on the reference dataset and assigns the reads that are highly matched with the reference sequences to the filtered dataset of a specific target gene (`specific filtered dataset`). If the file size of `specific filtered dataset` exceeds 10MB, GeneMiner will save it in `big_reads` and refilter.

4.3.3 assembled_out

`assembled_out` <folder>:

Used to store assembled contigs.GeneMiner assembles contigs using `specific filtered datasets`. Depending on the completion of the assembly, it can also be subdivided into `short_contig` and `contig` under `assembled_out`. Compared the recovered target gene with the average length of the reference sequence, the contig with a ratio greater than `-limit_min_ratio` will be put into the `contig` folder, and the contig with a ratio less than `-limit_min_ratio` will be put into the `short_contig` folder.

4.3.4 GM_results

`GM_results` <folder>:

used to store all the target genes mined by GeneMiner. `GM_results` is one of the most important folders.

4.3.5 bootstrap_out

`bootstrap_out` <folder>: Used to store the evaluation records.GeneMiner resamples `GM_results` based on the base substitution model and performs iterative checks on `GM_results`. The `bootstrap_out` folder contains `reference_database`, `filtered_out`, `assembled_out`, `GM_results`, `high_quality_results` and `bootstrap.csv`

NOTE: `bootstrap_out` is only generated when the `-bn` / `--bootstrap` option is used

(1) `reference_database` <folder>:

Used to store the mutated reference sequences . The target gene's initial acquisition is designated as Seq1, and its variation rate (v) is calculated by comparing it with the reference sequences. Combine the base substitution model built from the base composition of the reference sequences to resample the recovered target sequence with v to obtain ref_1, ref_2 ... ref_n

(2) `filtered_out` <folder>:

Used as the new reference to re-run GeneMiner to obtain the new filtered dataset.

(3) `assembled_out` <folder>:

GeneMiner completes the assembly using the new filtered dataset and stores the new assembly results

(4) `GM_results` <folder>:

Used to store all the new recovered target genes.

(5) `high_quality_results` <folder>:

GeneMiner compares the target genes in `GM_results` and `bootstrap_out/GM_results`, and gives a score for each target gene.

The target genes with score higher than 99 will be stored in the high_quality_results folder.

(6) `bootstrap.csv` <file>:

Assessment record sheet

4.3.6 results.csv

`results.csv` <file>:

Record various information about the target genes.

The columns in this file, which are separated by commas, provide different details about each retrieved target gene. It is simple to import the file into applications like Excel. This table's column headings are described (from left to right) as follows:

Column	Description
gene	target gene's name
k1	Length of kmer for filtering reads
re_k1	Length of kmer for filtering reads after re-filtering
richness	Approximate sequencing depth based on the target gene's filtered dataset
limit	Limit of the k-mer count
seed	The starting sequence used by GeneMiner in assembling contigs
k2	Length of kmer for assembling reads
ref_length	Average length of the reference sequences
short_contig_length	Length of the short contig

Column	Description
contig_length	Length of the contig
scaffold_length	Length of the scaffold
bootstrap_number	Number of resampling
score	target gene assessment score (bootstrap score)

NOTE: `bootstrap_number` and `score` are only printed when the `-bn` / `--bootstrap` option is used

5. Example

5.1 Mining chloroplast genes from genome skimming data

GeneMiner can recover almost all chloroplast (cp) genes from genome skimming data with sufficient coverage (>10x) and closely related reference sequences.

```
#Mining single cp gene : matk
geneminer.py -1 skimming_data1.fq.gz -2 skimming_data2.fq.gz -rtfa
shallow_ref/matK.fasta -t 4 -o matK_out
#Mining multiple cp genes
geneminer.py -1 skimming_data1.fq.gz -2 skimming_data2.fq.gz -rtfa Ref_cp/ -t 4
-o cp_out1
#Mining all cp genes
geneminer.py -1 skimming_data1.fq.gz -2 skimming_data2.fq.gz -rtgb chloro.gb -t
auto -b 300 -min 300 -max 5000 -o cp_out2
```

5.2 Mining mitochondrial genes from genome skimming

Mitochondrial (mito) genes are often more difficult to recover than chloroplast genes, generally because (1) mitochondrial genes are more variable (2) the raw data does not contain too many mitochondrial genes. In this case, you can increase the size of the raw data and choose more closely related reference sequences.

```
#Mining mito genes
geneminer.py -1 skimming_data1.fq.gz -2 skimming_data2.fq.gz -rtgb mito.gb -b
300 -min 300 -max 5000 -o mito_out
```

5.3 Mining nuclear genes from genome skimming

GeneMiner can mine highly repetitive regions in the nuclear genome (e.g. nrDNA).

```
#mining ITS
geneminer.py -1 skimming_data1.fq.gz -2 skimming_data2.fq.gz -t 4 -rtfa
shallow_ref/ITS.fasta -b 75 -o ITS_out
```

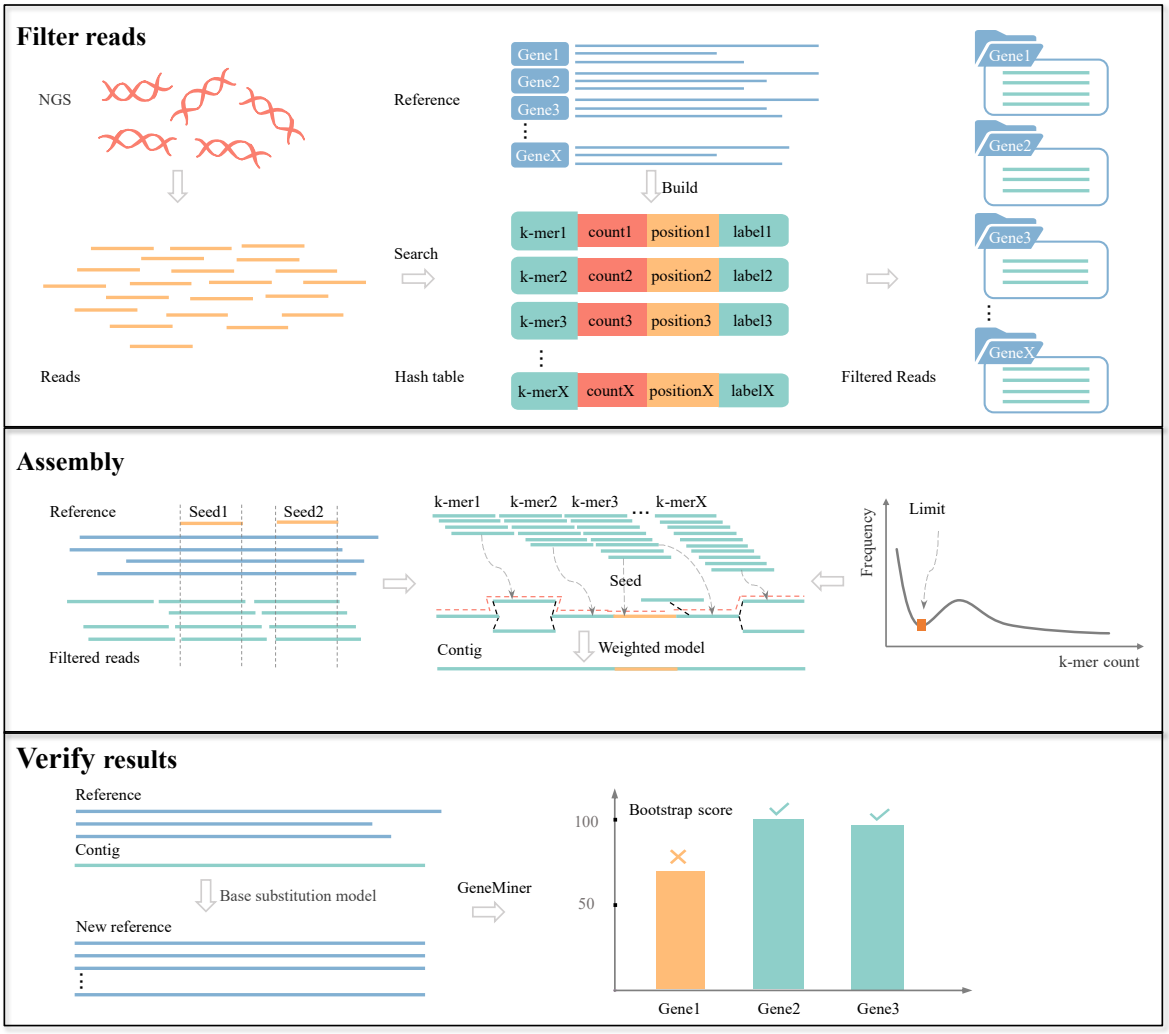
5.4 Mining Angiosperms353 genes from transcriptome data

Angiosperms353 are 353 putatively single-copy protein-coding genes identified by the One Thousand Plant Transcriptomes Initiative to design a set of targeted sequencing probes for phylogenetic studies of any angiosperm group.

```
#Mining Angiosperm 353 genes
geneminer.py -1 Arabidopsis_thaliana_sim1.fq.gz -2
Arabidopsis_thaliana_sim2.fq.gz -rtfa ref_Arabidopsis_353 -k1 29 -k2 41 -t 4 -
limit_count auto -b 10000 -o Angiosperm353
```

6. Methods

There are three steps in the primary workflow. First, reliable reads were obtained based on the reference of closely related taxa using a k-mer filtering method; Then, extend contigs from self-discovery seeds based on the weighted de Bruijn Graph. Finally, a base substitution model and repetitive sampling were used to statistically evaluate the impact of the reference on the assembly and confirm the assembly results.



6.1 Filtering reads

GeneMiner filters reads with high similarity to the reference of closely related taxa. The process consists of two stages: building the reference hash table and retrieving the reference hash table. In creating the reference hash table, we split the reference sequences into k-mers (k-mer is the iterative division of the reads into subsequences containing k bases) and the record position information, the occurrence number, and the corresponding labels of these k-mers. Finally, GeneMiner stored these states into the reference hash table. The total number of k-mers is

$$\sum_{i=1}^{n} L_i - k_f + 1$$

where L_i is the length of the i-th reference sequence, n is the total number of reference sequences, and k_f is set by the user. In retrieving the reference hash table, the raw input is next-generation sequencing (from Illumina, Roche-454, ABI, or other sequencing platforms) file in FASTQ format.

Each read in the raw input is also split into k-mers with the total number (L-k_f+1), where L is the read length and k_f is set as before. If the k-mer of a read appears in the reference hash table, the read is reserved and assigned to the target gene's filtered dataset with labels.

6.2 Assembly

Geneminer developed a seed self-discovery and extension algorithm based on the weighted de Bruijn Graph. The general flow is as follows:

(I) Make k-mers. Split the filtered reads into k-mers and build the k-mers set recorded as *T*. The k_a used in the assembly may differ from the k_f used in the filtering.

(II) Remove low-quality k-mers. The program will automatically fit the k-mer frequency distribution of set *T*, assign each target gene a minimum k-mer frequency threshold (*limit*) based on the bottom of the first L-peak, and eliminate any k-mers with frequencies below the *limit*.

(III)Choose seed. Split the reference sequences into k-mers using k_a and build another k-mers set recorded as *R*. If the high-frequency k-mer also appears in the set *T*, it will be regarded as a supposed conservative region and preserved as a candidate seed.

The candidate seed weighted score is calculated as

$$W_{seed} = C_{read} * log_{10}(C_{ref} + 1)$$

where C_{read} represents the occurrence number of the k-mer in the set *T*, C_{ref} represents the occurrence number of the k-mer in the set *R*. GeneMiner will automatically replace the candidate seed to achieve the optimal assembly, and the candidate seed with high weighted score will be prioritized.

(IV) Seed extension. Take each k-mer as a node and assign a weighted score according to its frequency and position in the reference sequences. The node weighted score is calculated as

$$W_{node} = C_{read} * e^{1-|P_1-P_2|}$$

where C_{read} represents the occurrences of the k-mer in the set *T*, P_1 represents the current assembly position of the k-mer, and P_2 represents the average position of the k-mer in the set *R*. Using the candidate seed as starting node, search the set *T* to find nodes with k-1 overlapping bases as adjacent nodes, where the nodes with high weighted scores will be prioritized. Connect the neighbouring nodes until no neighbouring nodes can be found. Finally, the overlapping cluster (contig) with the highest accumulated weighted score will be the output.

6.3 Verifying results

Based on the base substitution model and repeated sampling, GeneMiner creatively suggests a verification approach that can statistically assess the influence of the reference on the assembly results and confirm the stability of the assembly results.

(I) The target gene's initial acquisition is designated as Seq_1, and its variation rate (*v*) is calculated by comparing it with the reference sequences.

(II) Combine the base substitution model built from the base composition of the reference sequences to resample the recovered target sequence with *v* to obtain ref_1, ref_2 ... ref_n

(III) Use ref_1, ref_2 ... ref_n as the new reference to re-run the whole process and get the new recovered target sequence noted as Seq_2

(IV) Seq_1 and Seq_2 are compared in the double sequence alignment, and then the identity (called *bootstrap score* in GeneMiner) is used to assess the quality of the reference sequence and the assembly results. The identity is calculated as

$$bootstrap\ score = \frac{N}{L} * 100$$

where *N* represents the matched base number between Seq_1 and Seq_2, and *L* represents the alignment length.

7. Contact

Please check Geneminer's homepage first. If your question is running specific,please do not be surprised and report it to us. We usually have quick response to bugs.

- Find Questions & Answers at GeneMiner Discussions: **Recommended**
- Report Bugs & Issues at GeneMiner Issues: Please avoid repetitive or irrelevant issues
- QQ group (ID: 78266311): Mainly for mutual help, responses are likely to be not timely

DO NOT directly write to us with your questions, instead please post the questions **publicly**, using above platforms Our emails (xiepulin@scu.edu.cn, 1791173948@qq.com) are only for receiving public question alert and private data (if applied) associated with those public questions. When you send your private data to us, enclose the email with a link where you posted the question. Our only reply emails will be a receiving confirmation, while our answers will be posted in a public place.

8. Citation

When you use GeneMiner please cite:

GeneMiner : a software for extracting phylogenetic markers from next-generation sequencing data