

GeneMiner2 Manual

2025-8-22

Version: 20250709

Release Date: July 2025

Maintainer: Evolutionary Computing & Phylogenomics Lab, SCU

Contact: Xinyi_Yu2021@163.com

Repository: <https://github.com/sculab/GeneMiner2>

GeneMiner2 is a comprehensive software toolkit developed for extracting, assembling, and analyzing phylogenetic markers from a wide range of genome sequencing data, including genome skimming and transcriptome sequencing datasets. Designed with flexibility and usability in mind, it supports both graphical and command-line interfaces, allowing users to process a wide range of input data with minimal preprocessing. The toolkit implements workflows for extracting single-copy orthologous genes, recovering complete or partial organelle sequences, and reconstructing phylogenetic trees using automated pipelines. Its core algorithms are optimized for low-coverage sequencing data, enabling effective marker recovery even in non-model plant taxa where reference genomes are incomplete or unavailable. GeneMiner2 integrates multiple stages of analysis, including sequence filtering, de novo assembly, ortholog detection, alignment, trimming, and phylogenetic inference. It supports both individual and batch mode analysis, making it suitable for large-scale studies involving dozens or hundreds of taxa.

Table of Contents

| | |
|--|----|
| A Rough Guide to GeneMiner2..... | 1 |
| Table of Contents..... | 2 |
| 1. Introduction | 4 |
| 1.1 Overview | 4 |
| 1.2 Citation..... | 5 |
| 2. Installing..... | 6 |
| 2.1 Windows..... | 6 |
| 2.2 Linux | 6 |
| 2.3 macOS | 8 |
| 3. A Tour of GeneMiner2..... | 9 |
| 3.1 Graphical interface | 9 |
| 3.2 Command-line interface | 17 |
| 4 Directory Annotation..... | 18 |
| 4.1 Assemble results..... | 18 |
| 4.2 Genome assembly | 19 |
| 4.3 Batch- analysis results..... | 19 |
| 4.4 Other Files | 19 |
| 5 Function and graphical interface introduction..... | 20 |
| 5.1 Graphical interface | 20 |
| 5.2 Function Menu | 22 |
| 6 Algorithm Overview..... | 32 |
| 6.1 Fast read filtering | 32 |
| 6.2 Fine-grained read selection | 35 |
| 6.3 Adaptive k -mer selection..... | 39 |
| 6.4 Prameters calculation and settings | 40 |

| | |
|-------------------------------|----|
| 6.5 Trimming Strategies | 41 |
| 7 FAQ | 41 |
| Changelog..... | 43 |
| References | 44 |

1. Introduction

1.1 Overview

GeneMiner2 is an automated pipeline from sequence assembly to multi-gene alignment, phylogenetic tree reconstruction, divergence dating, and plastome recovery. It is accurate (even at as low as $5\times$ read depth), performant (more than $10\times$ faster than whole-genome assembly), and flexible, empowering studies across a wide range of taxa, from Lamiaceae to Liliaceae. It features an easy-to-use graphical user interface on Windows and macOS platforms and bundles several convenience features, including a parameter calculator and an integrated phylogenetic tree viewer.

GeneMiner2 integrates a wide range of bioinformatic software, including MAFFT, MUSCLE, NCBI BLAST+, NOVOPlasty, PAML and others. Intermediate and final results for each analysis are kept, enabling flexible downstream use. GeneMiner2 can automatically retrieve Angiosperms353 genes, chloroplast genomes and mitochondrial genomes through an on-line database. We kindly ask you to cite GeneMiner2 and other database and software involved in your analysis.

Bundled Software

- **Sequence Similarity Search:**
 - NCBI BLAST+ v2.16 (Camacho et al., 2009)
- **Read Mapping:**
 - Magic-BLAST v1.7.2 (Boratyn et al., 2019)
 - Minimap2 v2.1 (Li, 2018)
- **Multiple Sequence Alignment:**
 - MAFFT v7.52 (Kato et al., 2002)
 - MUSCLE v5.3 (Edgar, 2022)
- **Alignment Trimming:**
 - trimAl v1.4 (Capella-Gutierrez et al., 2009)
- **Gene Tree and Species Tree Reconstruction:**
 - FastTree v2.1
 - ASTRAL-III (Zhang et al., 2018)
- **Orthology Inference:**
 - OrthoFinder v2.5.5 (Emms & Kelly, 2019)
- **Divergence Time Estimation:**
 - MCMCTree (part of the PAML 4 package; Yang, 2007)
- **Plastome Assembly:**
 - NOVOPlasty v4.3.4 (Dierckxsens et al., 2017)

1.2 Citation

Tools

Boratyn GM, Thierry-Mieg J, Thierry-Mieg D, *et al.* Magic-BLAST, an accurate RNA-seq aligner for long and short reads. *BMC Bioinformatics*. 2009;20:405.

Camacho C, Coulouris G, Avagyan V, *et al.* BLAST+: architecture and applications. *BMC Bioinformatics*. 2009;10:421.

Capella-Gutiérrez S, Silla-Martínez JM, Gabaldón T. trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses. *Bioinformatics*. 2009;25(15):1972–1973.

Dierckxsens N, Mardulyn P, Smits G. NOVOPlasty: de novo assembly of organelle genomes from whole genome data. *Nucleic Acids Research*. 2017;45(4):e18.

Edgar RC. Muscle5: High-accuracy alignment ensembles enable unbiased assessments of sequence homology and phylogeny. *Nature Communications*. 2022;16(6968).

Emms DM, Kelly S. OrthoFinder: phylogenetic orthology inference for comparative genomics. *Genome Biology*. 2019;20(1):238.

Heng L. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*. 2018;34(18):3094–3100.

Katoh K, Standley DM. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molecular Biology and Evolution*. 2013;30(4):772–780.

Price MN, Dehal PS, Arkin AP. FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments. *PLOS ONE*. 2010;5(3):e9490.

Yang Z. PAML 4: phylogenetic analysis by maximum likelihood. *Molecular Biology and Evolution*. 2007;24:1586–1591.

Zhang C, Rabiee M, Sayyari E, Mirarab S. ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics*. 2018;19:153.

Emhash

<https://github.com/ktprime/emhash>

Quotient Hash Tables

Géraud R, Lombard-Platet M, Naccache D. Quotient hash tables: efficiently detecting duplicates in streaming data. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19)* (ACM, 2019).

2. Installing

2.1 Windows

Download GeneMiner2 from SourceForge and extract the package to a local directory on your hard drive. **Avoid using Chinese or non-ASCII characters in the folder path**, and do not place the software on removable or network drives.

To launch the graphical user interface, simply double-click on **GeneMiner.exe**. No installation is required.

- It is recommended to create a new, empty output folder for each run to avoid overwriting or deleting previous results.
- **Parallel execution notice:** Do not run multiple windows simultaneously within the same folder. If you wish to launch a second instance of GeneMiner2, **duplicate the entire GeneMiner directory and execute the second window in that separate copy**. This ensures that temporary files and outputs are properly isolated and prevents data corruption.

2.2 Linux

Although GeneMiner2 has native support for Linux and other Unix-like operating systems, we recommend desktop Linux users to run Windows binaries, which provide multiple convenience tools and simplify phylogenetic analyses. For large-scale analysis, you can build GeneMiner2 natively as documented further below.

Running Windows binaries with Steam

1. Install [Steam](#) and create an account.
2. Download and extract the GeneMiner2 package.
3. In Steam, click **Add a Non-Steam Game** and select **GeneMiner.exe**.
4. In Library, right-click the entry → **Properties** → **Compatibility**.
5. Enable **Force the use of a specific Steam Play compatibility tool** and select **Proton 9.0-4**.
6. Click **Play** to launch the program with GUI support.

Surprisingly, this method is by far the most efficient, friendly and stable solution to run GeneMiner2 GUI on Linux.

Running Windows binaries with pre-installed Wine

Due to Proton not being available in many environments, you might need to run GeneMiner

with an open-source Wine management solution like umu-launcher or directly using pre-installed Wine:

WINEPREFIX=~/.wine umu-run ~/geneminer2/GeneMiner.exe

WINEPREFIX=~/.wine wine ~/geneminer2/GeneMiner.exe

Modify the paths to match your actual directory structure.

Running native binaries with a command-line interface

GeneMiner2 provides a native command-line version for Unix-like systems, including Linux, BSDs and macOS. This version is ideal for high-throughput or automated phylogenetic analysis.

1. Install the default C++ compiler and zlib. Advanced users can install zlib-ng or cloudflare-zlib in place of zlib to improve performance. On Ubuntu 20.04 and later, this can be done as follows. Other users should install them with the system package manager.

```
sudo apt install build-essential zlib1g zlib1g-dev
```

2. Install Haxe 4 compiler and configure the C++ target. Users on Ubuntu 20.04 and later can execute the following commands. Other users can refer to Haxe website for installation instructions.

```
sudo apt install haxe
haxelib setup ~/haxelib
haxelib install hxcpp
```

3. Install Python dependencies. If conda is available, this can be done using the following commands. Otherwise, please install these packages manually.

```
conda create -c conda-forge -n geneminer python=3.11 numpy=2.1.3 biopython matplotlib
pip install scipy
conda activate geneminer
```

4. Unpack the source code and build the binaries:

```
cd GeneMiner2
make
```

5. Install run-time dependencies. If conda is available, run the following command. Otherwise, install them manually.

```
conda install -c bioconda aster blast clustalo fasttree iqtree mafft magicblast minimap2
muscle raxml-ng trimal veryfasttree
```

6. Run `cli/geneminer2` to ensure a working build.

The command-line interface requires a tab-delimited table of samples.

For paired-end reads, include three columns: species name, Read1 path, and Read2 path.

For single-end reads, include two columns: species name and Read1 path.

Example table:

| | | |
|-----|----------------------|----------------------|
| Sp1 | /path/to/Sp1_1.fq.gz | /path/to/Sp1_2.fq.gz |
| Sp2 | /path/to/Sp2_1.fq.gz | /path/to/Sp2_2.fq.gz |
| Sp3 | /path/to/Sp3.fq.gz | |

Run the main pipeline:

```
cli/geneminer2 -f /path/to/samples.tsv -r /path/to/ref_dir -o output_dir
```

To view available command-line options:

```
cli/geneminer2 -h
```

To customize processing steps, add specific parameters:

```
cli/geneminer2 trim combine tree -f samples.tsv -r angiosperm353 -o output_dir -ts  
consensus -tm all -tr 0.5 -cd 0.2 -cn 5 --msa-program muscle
```

The resulting trees will be placed at `output_dir/Coalescent.tree` or `output_dir/Concatenation.tree`, depending on the value of `-m` parameter.

Notes:

All parameters are consistent with the GUI version. For proportion-based arguments, use decimal format (e.g., **0.5** for 50%).

Advanced users may use hidden flags such as **--min-depth** to fine-tune sequence retention and filtering behavior.

2.3 macOS

GeneMiner2 has a Wineskin-based GUI package that works on macOS, which is also hosted on Sourceforge. If you meet an error like "GeneMiner.app is damaged", run this in terminal:

```
xattr -cr /Applications/GeneMiner.app
```

The macOS package is not fully tested and may be less efficient than the Windows version. Hence, it is not recommended to run GeneMiner2 GUI on macOS for large-scale analysis. Alternatively, follow the instructions in the last section to obtain a native command-line version of GeneMiner2.

3. A Tour of GeneMiner2

3.1 Graphical interface

3.1.1 Single sample (Chloroplast genes assembly, FASTA references)

Loading the data files

- **Sequencing Data** (DEMO/DEMO1/DATA/seq): Click [**File > Load Sequencing Files**].

Select the sequencing data in .gz format.

(For batch processing with **multiple samples**, see **Demo 3 (DEMO3/DEMO3.md)** for details.)

- **Reference Sequences** (DEMO/DEMO1/DATA/A_lyrata): Click [**File > Load Reference**].

Select FASTA reference sequences.

If no reference file is available locally, you can download standard references by selecting [**Files > Download References**] from the top menu. Available datasets include:

- Plant Chloroplast Genome
- Plant Mitochondrial Genome
- Animal Mitochondrial Genome

(Note: Select Download as single gene if you want to extract genes.)

- Angiosperms 353 Genes

Calculating Parameters

Click [**Tools > Calculate Parameters**] .

Calculate Parameters

Input

Read Length 150 Max Variation 0.1

Mean Depth 10 Mean Gene Size 1000

Sample Type Genome Reference Type Transcriptome

Calculate

Result

Filter K 31 Error Thr. 1

Step Size 1 Search Depth 4096

Retent. Thr. 0 Boundary Unlimited

Max Diff. 0.1 Trim Method All Fragments

Apply Close

Click **Calculate** to estimate , and then click **Apply** to finalize the settings.

This demo uses the default parameters.

Obtain Genes

Click [**Analysis > Filter & Assemble**] to obtain Genes.

NOTE: Do not manually close the command line window; it will close automatically once the process is complete.

NOTE: For importing multiple pairs of sequencing files, select [**Batch > Filter & Assemble**] to extract.

This demo uses the default parameters.

Basic Option

Filter

☐ Reads/File (M) 10

Filter K Value: 31

Filter Step Size: 4

☒ High-Speed

Further Filtering

Depth Limit: 768

File Size Limit: 6

Assemble

☒ Auto-Estimate K

21 -> 51

Fixed K: 39

Err. Threshold: 2

Boundary: Auto

Search Depth: 4096

启用此选项将大部分数据存储在内存中，而不是实时计算。相对于禁用该选项可以使处理速度几乎加倍，但RAM的使用量也会加倍。

☐ Hide Terminal Window

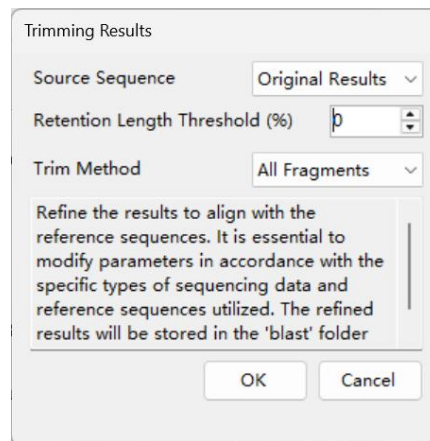
OK Cancel

Click [**Open**] to view the results located in the 'results' folder.

Data Cleaning

Click [**Analysis > Trim With Reference**].

NOTE: Parameters was calculated by "Calculate Parameters".



Click [**Open**] to view the results located in the 'blast' folder.

3.1.2 Single sample (Chloroplast genes assembly, GenBank references)

Loading the data files

- **Sequencing Data** (DEMO/DEMO2/DATA/PLANT/GENE): Click [**File > Load Sequencing Files**]. Select the sequencing data in .gz format.

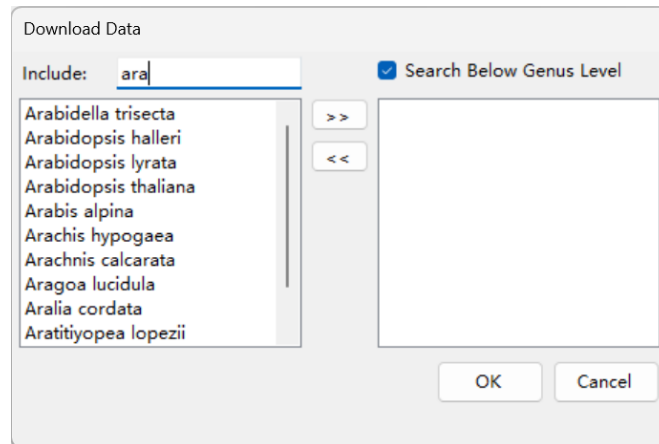
- **Reference Sequences** (DEMO/DEMO2/DATA/ref/OK166971.gb): Click [**File > Load Reference**]. Select GenBank (.gb) reference sequences.

If no reference file is available locally, you can download standard references by selecting [**Files > Download References**] from the top menu. Available datasets include:

- Plant Chloroplast Genome
- Plant Mitochondrial Genome
- Animal Mitochondrial Genome

(Note: Select Download as single gene if you want to extract genes.)

- Angiosperms 353 Genes



Calculating parameters

Click [**Tools > Calculate Parameters**] to estimate assembly thresholds based on the input data.

In the pop-up window, click **Calculate** to preview estimated values, and then click **Apply** to confirm the suggested parameter settings.

This demo uses the default parameters.

Assemble

Click [**Analysis > Filter & Assemble**] to initiate reference-based gene extraction.

GeneMiner2 will:

- Automatically detect whether the input is paired-end or single-end sequencing data.
- Use a hash table-based high-throughput filtering algorithm to process large-scale FASTQ data (>120 Mb/s), enabling efficient read screening even at low sequencing depths.

The screenshot shows the 'Basic Option' dialog box in GeneMiner2. It is divided into several sections: 'Filter' with options for 'Reads/File (M)' (10), 'Filter K Value' (31), 'Filter Step Size' (4), and a checked 'High-Speed' option; 'Further Filtering' with 'Depth Limit' (768) and 'File Size Limit' (6); 'Assemble' with a checked 'Auto-Estimate K' option, a range of 21 to 51, 'Fixed K' (39), 'Err. Threshold' (2), 'Boundary' (Auto), and 'Search Depth' (4096); and a large text area on the right containing Chinese text: '启用此选项将大部分数据存储在内存中，而不是实时计算。相对于禁用该选项可以使处理速度几乎加倍，但RAM的使用量也会加倍。'. At the bottom, there is a 'Hide Terminal Window' checkbox and 'OK' and 'Cancel' buttons.

Trim low-quality regions

Click [Analysis > Trim With Reference] to remove low-quality regions and sequences.

3.1.3 Single sample (Chloroplast genome assembly)

Loading the data files

- **Sequencing Data:** Click [File > Load Sequencing Files]. Select the sequencing data.

Assemble

Use [Analysis > Plant Chloroplast Genome] to perform *de novo* organelle genome assembly and annotation. **This part uses the following parameter settings. "Reads/File (M)" is set to 10.**

Basic Option

Filter

☒ Reads/File (M) 10

Filter K Value: 17

Filter Step Size: 4

☒ High-Speed

Further Filtering

Depth Limit: 768

File Size Limit: 6

Assemble

☒ Auto-Estimate K

21 -> 51

Fixed K: 39

Err. Threshold: 2

Boundary: Auto

Search Depth: 4096

The Filter K Value (kf) depends on three factors: acquisition rate (p) of gene-specific reads, read length (r), and variation degree (v) between reference and target sequences. Typically, with p at 0.99, r at 150, and v at 0.1, kf is around 31, our default setting. If p is 0.95 and v increases to 0.2 (r stays at 150), kf drops to 17, the proposed minimum threshold.

☐ Hide Terminal Window

OK Cancel

Click [OK] to proceed with the default parameters for assembly.

Download Data

Include: ara

☒ Search Below Genus Level

Arabidella

Arabidopsis

Arabis

Arachis

Aragoa

Aralia

Arapatiella

>>

<<

Arabidopsis

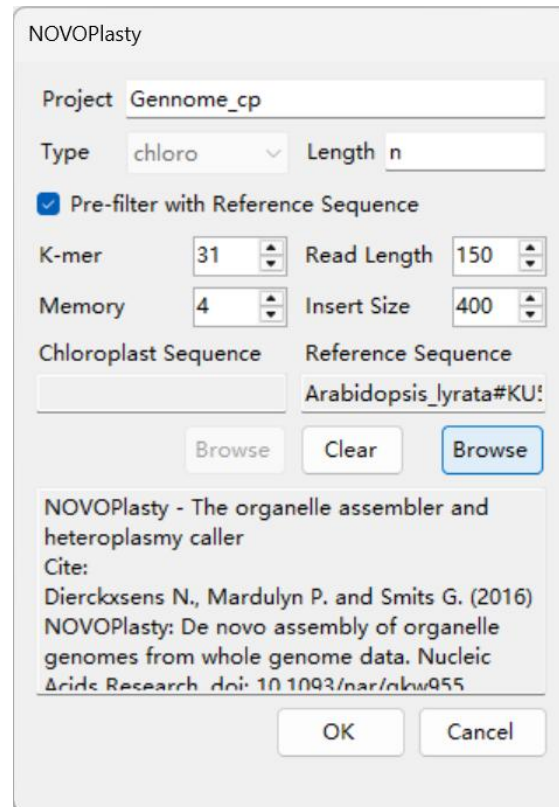
OK Cancel

Select closely related taxa to retrieve their reference genomes.

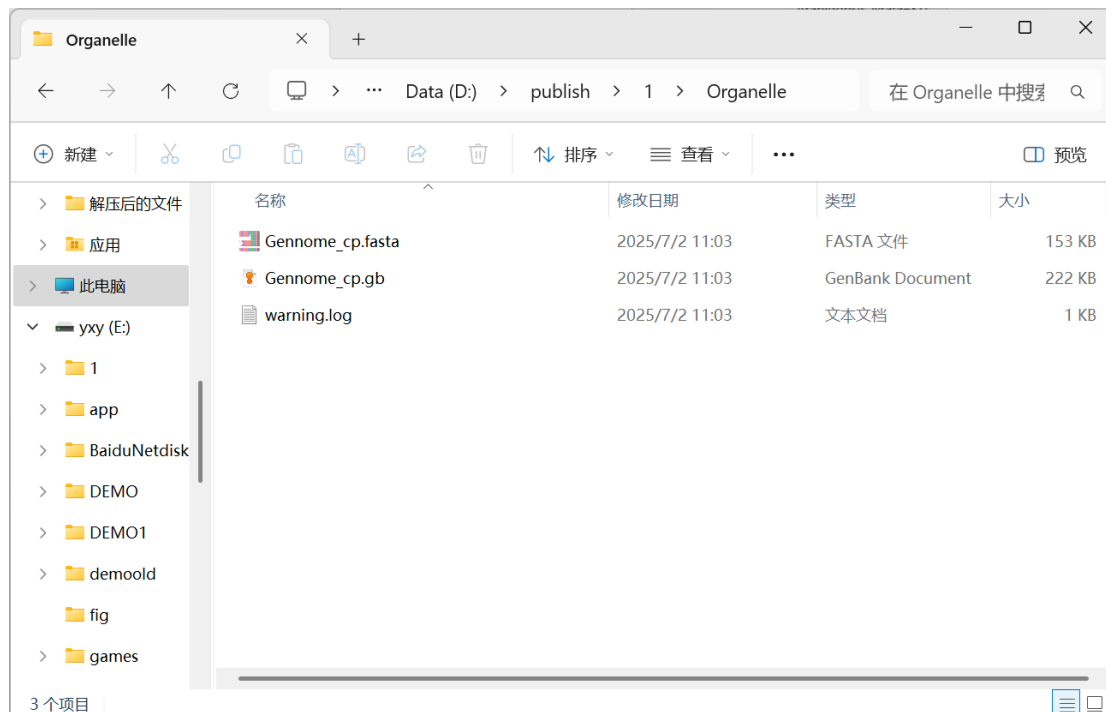
NOTE: For importing multiple samples, select [Batch > Plant Chloroplast Genome] to extract.

Insufficient memory may cause assembly failure. Consider increasing memory if needed.

Set the *Memory* to 4.



Click [**Open**] to view the results located in the 'Organelle' folder named 'Genome_cp.fasta'.



3.1.4 Batch samples (SCGs, entire pipeline)

1. Data Preparation

- **Sequencing Data (DEMO3/DATA/PLANT/)**: Load second-generation sequencing files [**File > Load Sequencing Files**].
- **Transcript Data (DEMO/DEMO3/DATA/Phytozome)**: Load reference transcript sequences [**File > Load References**], typically derived from closely related species. These transcripts will serve as the basis for identifying single-copy gene candidates.

2. Identify single-copy gene references

Click [**Analysis > Find Single Copy Genes**] to scan the transcript reference and extract genes present as a single ortholog across the dataset.

These SCGs will serve as the working reference for downstream extraction.

3. Calculate Parameters (see 3.1.1), this demo uses the default parameters.

4. Filter and assemble single-copy genes

- Click [**Change**] to specify the result output directory.
- Ensure that the previously generated SCG reference is selected via [**File > Load References**].
- Use [**Batch > Filter & Assemble**] to assemble target regions.

Assembled gene sequences are available in the 'results' directory.

Basic Option

Filter

☐ Reads/File (M) 10

Filter K Value: 31

Filter Step Size: 4

☒ High-Speed

Further Filtering

Depth Limit: 768

File Size Limit: 6

Assemble

☒ Auto-Estimate K

21 -> 51

Fixed K: 39

Err. Threshold: 2

Boundary: Auto

Search Depth: 4096

启用此选项将大部分数据存储在内存中，而不是实时计算。相对于禁用该选项可以使处理速度几乎加倍，但RAM的使用量也会加倍。

☐ Hide Terminal Window

OK Cancel

Note: Do not manually close the command line window during batch execution; it will close automatically upon completion.

5. Trim low-quality regions

Click [**Batch > Trim With Reference**] to remove low-quality regions and sequences.

Trimmed results can be previewed in the 'blast' directory.

6. Combine aligned sequences

Click [**Batch > Combine Results**] to merge and filter aligned genes across samples. (Parameters were set using Calculate Parameters.)

Concatenated gene alignments are saved as:

- combined_results.fasta: All aligned SCGs.
- combined_trimmed.fasta: SCGs after quality trimming.

The per-gene results are stored in 'combined_results' and 'combined_trimmed' directories.

7. Build phylogenetic tree

- Click [**Batch > Build Phylogenetic Tree**] to initiate tree construction.
- Select the matrix type and tree-building method, enable bootstrap. Choose whether to build a rooted tree, specify the outgroup, and decide whether to perform tree time calibration.
- The output files include Coalescent.tree and Concatenation.tree.

8. Tree time calibration

- Double-click a branch in the tree to edit its divergence time.

3.2 Command-line interface

3.2.1 Single sample (Chloroplast genes assembly, FASTA references)

Run the following command. The results will be placed at DEMO/DEMO1/results/1_Arabidopsis_thaliana/results.

```
cli/geneminer2 filter refilter assemble -r DEMO/DEMO1/DATA/A_lyrata  
-f DEMO/DEMO1/samples.tsv -o DEMO/DEMO1/results
```

If the command does not work, try modifying DEMO/DEMO1/samples.tsv to use absolute paths to sequencing data.

NOTE: The command-line interface does not contain convenience features like importing reference sequences from GenBank files. Please use the graphical interface or other utilities to convert them to FASTA files.

3.2.2 Batch samples (SCGs, entire pipeline)

Install OrthoFinder separately and run on the data given by DEMO3.

```
orthofinder -d -f DEMO/DEMO3/DATA/Phytozome/
```

Copy the resulting single copy genes to a convenient location, for example DEMO/DEMO3/SCG. Run GeneMiner2 as follows:

```
cli/geneminer2 -r DEMO/DEMO3/SCG -f DEMO/DEMO3/samples.tsv -o  
DEMO/DEMO3/results
```

The command-line version of GeneMiner2 runs the whole pipeline (excluding consensus) by default. The coalescent tree is placed at DEMO/DEMO3/results/Coalescent.tree. To build a concatenation tree, run:

```
cli/geneminer2 tree -r DEMO/DEMO3/SCG -f DEMO/DEMO3/samples.tsv -o  
DEMO/DEMO3/results -m concatenation
```

Note the action 'tree' given right after the GeneMiner2 executable. If one or more action is given, GeneMiner2 CLI will only run the given steps. It requires dependent steps to have fully completed, which must be managed carefully. All command-line switches can be listed using cli/geneminer2 -h. They fully correspond to their counterparts in the Windows version but allow extra flexibility in large-scale analysis.

NOTE: Calculate Parameters is another missing convenience feature in the command-line interface. It is recommended to determine the parameters using the GUI version first, or alternatively, using theoretical calculation. To replicate an analysis by the Windows version, specify **every** parameter using its respective value shown in the GUI.

4 Directory Annotation

4.1 Assemble results

results:The sequence with the highest weight in the assembly results, the final result.

- kmer_dict_k31.dict: kmer dictionary file, formatted as: kmer fragment (in hexadecimal), kmer count (in hexadecimal).

- result_dict.txt: Results file, formatted as: gene name, sequence assembly status, number of

reads assembled.

- ref_reads_count_dict.txt: Total number of kmers split from each reference gene sequence.

- result_dict.txt: Results file, formatted as: gene name, sequence assembly status, number of reads assembled.

4.2 Genome assembly

Organelle: Assembly results of the organelle genomes.

- Genome_cp.fasta: Plant chloroplast genome assembly result.

- Genome_cp.gb: Annotated plant chloroplast genome assembly result.

- Genome_mito_plant.fasta: Plant mitochondrial genome assembly result.

- temp: Files from the organelle genome assembly process that were not completed due to terminal closure.

- Genome_mito.fasta: Animal mitochondrial genome assembly result.

4.3 Batch- analysis results

Your Sequencing Filename: A folder named after the sequencing sequence, storing the assembly results obtained from each sequencing sequence separately.

combined_results: Stores the combined result files.

combined_trimmed: Stores the result files after merging and trimming.

combined_results.fasta: Concatenated result file.

combined_trimmed.fasta: Trimmed concatenated result file.

aligned: Results of multiple sequence alignment.

4.4 Other Files

contigs_all: All possible assembly results.

filtered: The fq files obtained after filtering.

iteration: Files obtained from the first or multiple iterations, with filenames and meanings consistent with the parent folder.

large_files: Original fq files that exceed the depth or file size limits during further filtering. If all filtering results are within limits, this folder will not appear.

log.txt: Log file.

consensus: Maps the result sequences to the filtered fq files. Sequences with a number of ambiguous bases above the set value will be retained.

supercontigs: Consensus reconstruction result files, generating degenerate sequences using IUPAC codes, with degenerate bases marking SNP sites.

paralogs: Result files from paralogous gene screening. The `_ref.fasta` file stores paralogous genes, the `csv` file records the number of times different base positions are mapped, and the `.pec.csv` file records the frequency of base mutations.

summary.csv: Summary statistics of the results, including:

- Reference Median Length: The median length of the reference sequences, used for filtering during the Reference-based Trimming step.
- Reads Counts: The number of reads retained after filtering and matching.
- Result Availability: Indicates whether an assembly result exists (1 = yes).
- Multicopy Presence: Indicates whether multi-copy sequences are detected (1 = yes)

PPD>result>supercontig>s8 _rm_paralogs> Final_kept_genes: This is the final result file of [Paralogous Detection (PPD)], for specific meanings of other files see [PPD GitHub](https://github.com/Bean061/putative_paralog#part2-trims-alignment-and-detectes-the-putative-paralogs).

5 Function and graphical interface introduction

5.1 Graphical interface

5.1.1 Sidebar

Output: The folder for saving results, defaulting to the results folder in the directory where the GeneMiner2 application is located.

Open: Opens the output directory in Windows Explorer.

Change: Select the folder where results will be saved. **Note:** The folder for saving results may be repeatedly emptied during operation, so be sure not to choose a folder containing important files. It is recommended to create a new folder for output with each analysis. If the same output folder is selected to continue previous analyses, opt not to clear the folder in subsequent analyses.

Threads: The number of threads is set to the total number of logical processors minus two by default. Users can adjust the thread count based on the number of available CPU cores on their machine.

5.1.2 References Column

Select: Whether to use this reference sequence.

ID: Identifier number of the reference gene.

Name: Name of the reference gene.

Ref. Count: Number of times the reference gene appears.

Ref. Length: Average length of the reference gene (in base pairs).

Reads: Number of reads matching the reference k-mers.

Assemble State/Count: Number of assembled sequences obtained(batch samples).

Ass. Length: Total length of the assembled sequence.

Max. Diff.: Maximum pairwise divergence between sequences from different species after consensus merging.

Assemble State: Status of sequence assembly(only single sample), which includes:

- **no filtered files:** The filtered result file for the gene does not exist.
- **no reads:** The filtered result file exists, but no usable read fragments could be extracted for assembly.
- **insufficient genomic kmers:** The number of genomic k-mer regions detected after filtering is insufficient for assembly.
- **no seed:** No valid seed sequences were identified to initiate assembly.
- **no contigs:** No contigs were generated due to insufficient read coverage or failed extension from seeds.
- **low quality:** Low accuracy of results, reads insufficient to cover assembled results.

Note: If any of the above filtering or assembly failures occur, it is recommended to adjust the reference, k-mer (kf,for filter),the filter step size , or error rate(for assemble).

- **success:** Assembly successful.

Ass. Length: Length of the assembly result.

Max.Diff.: Maximum divergence obtained from pairwise comparison of genes from different species after [Merge & Trim]. Right-click and select Max. Diff. to filter out low-quality result sequences based on maximum divergence, with a default filter set at 0.1. Once checked, selected sequences can be re-merged and trimmed for phylogenetic tree construction using the filtered and trimmed results.

5.1.3 Sequences Column

- **Select:** Whether to use this set of data files.

- **Data1:** Left end (end 1) of the sequencing file.

- **Data2:** Right end (end 2) of the sequencing file; if it is single-end sequencing, it automatically matches the content in Data1.

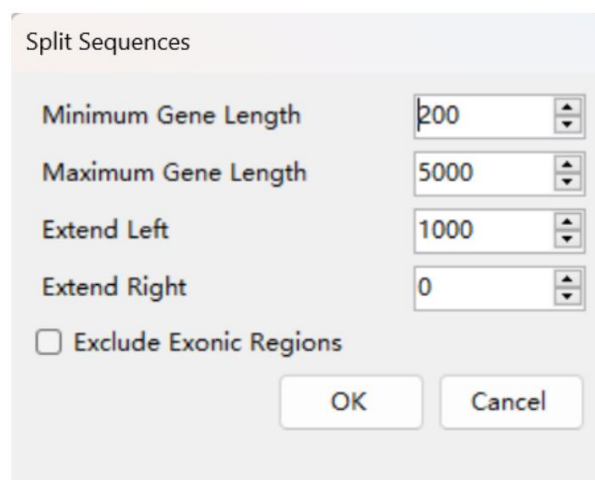
Note: The batch function supports sequencing data from multiple samples.

5.2 Function Menu

5.2.1 File

[File > Load References]: Choose reference sequence files in either fasta or genbank format; multiple reference sequence files can be selected at once.

If a genbank format file is selected, you can choose whether to import it as a gene list. If so, GeneMiner2 will automatically decompose the genes according to gene names, and the following settings dialog will pop up:



The image shows a dialog box titled "Split Sequences". It contains four input fields with numeric values and a checkbox. The fields are: "Minimum Gene Length" with value 200, "Maximum Gene Length" with value 5000, "Extend Left" with value 1000, and "Extend Right" with value 0. Each field has a small up/down arrow icon to its right. Below these fields is a checkbox labeled "Exclude Exonic Regions" which is currently unchecked. At the bottom right of the dialog are two buttons: "OK" and "Cancel".

| Parameter | Value |
|------------------------|--------------------------|
| Minimum Gene Length | 200 |
| Maximum Gene Length | 5000 |
| Extend Left | 1000 |
| Extend Right | 0 |
| Exclude Exonic Regions | <input type="checkbox"/> |

* Minimum Gene Length: The minimum length of genes to process

- * **Maximum Gene Length:** The maximum length of genes to process
- * **Extend Left/Right:** The length to extend on each side of a gene (the extended intron region length)
- * **Exclude Exonix Region :** Exon regions can be removed, retaining only the selected length of the intron regions.
- * **Note:** GeneMiner2 is designed to filter and assemble sequences that include exon regions; it is not recommended to directly import intron regions as reference.

[File > Load Sequencing Files]: Load second-generation sequencing data files, which can be in .gz or .fq format. For paired sequence files, select two files (even number) simultaneously to load. If only one file is selected, it will be loaded as single-end sequencing data.

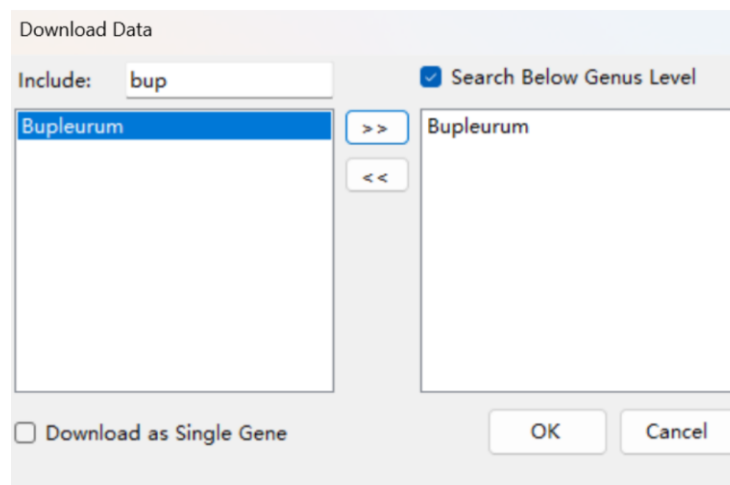
[File > Download References > Plant Chloroplast Genome]

[File > Download References > Plant Mitochondrial Genome]

[File > Download References > Animal Mitochondrial Genome]:

Retrieve organelle genomes from the software's local database for reference sequence import. The local database is populated with data from NCBI([NCBI](<https://www.ncbi.nlm.nih.gov/>)).

Enter the Latin taxonomic name of the genus or higher taxonomic rank in the input box, select the taxonomic group below, and click the >> button to add it to the list on the right.



- **Search Below Genus Level :** Only displays taxa related to the genus level. If unchecked, it defaults to showing taxa at the genus level and above. It is recommended to check this option.
- **Download as Single Gene:** Downloads the entire chloroplast genome sequence in fasta

format. If unchecked, it defaults to downloading the entire chloroplast genome in gb format, which will be split into multiple gene reference sequences for import.

Note: If the taxonomic group you are researching is not found, this means it is not available in the software's local database; please select a higher taxonomic rank as a substitute.

[File > Download Sequences > Angiosperms 353 Genes]: Retrieve the Angiosperms353 Gene Set (AGS) from the software's local database for reference sequence import. The local database sources data from the Kew Tree of Life Explorer ([Kew Tree of Life](<https://treeoflife.kew.org>)).

[File > Export References]: Choose an output folder to export the selected reference genes in fasta format.

[File > Export Sequencing Files]: Select an export folder, set the number of reads to skip, and the number of reads to preserve (export) in the popup dialog. For each pair of sequencing files, the exported files are named **.1.fq* and **.2.fq* (* is the sequence name).

[File > Export List]: Specify the filename and location to save the reference sequence information list in CSV format.

Note: If exporting within the same folder, change the names of previously exported sequences to avoid overwriting them.

[File > Save Project]: Save the data imported into the software in .geneminer format.

[File > Open Project]: Directly import previously loaded data (including references, result directories, and sequencing sequences).

5.2.2 Analysis

[Analysis > Filter > Filter from Scratch]: Use references to filter sequencing data, obtaining reads associated with target genes. The fq files of filtered results are saved in the 'filtered' folder within the output directory. If the filtering depth is too high or the files are too large, further filtering is recommended. Upon completion, the estimated depth of the filtered results will be displayed on the main interface list, and users can check the size of each gene's filtered file in the 'filtered' folder of the output directory.

[Analysis > Filter > Further Filter]: Perform further filtering on data that is too large or too deep in the filtered results. Original data that is too large or too deep is stored in the

'large_files' folder, and data after further filtering is saved in the 'filtered' folder.

[Analysis > Assemble]: Perform assembly using the filtered sequences, with the final results saved in the 'results' folder of the output directory.

[Analysis > Filter & Assemble]: Automatically complete all steps of filtering, (further filtering), and assembly using the current settings, with all results saved in the output directory.

Parameter settings will appear during the above steps, with specific meanings as follows:

Basic Option

Basic Option

Filter

☐ Reads/File (M) 10

Filter K Value: 31

Filter Step Size: 4

☒ High-Speed

Further Filtering

Depth Limit: 512

File Size Limit: 8

Assemble

☒ Auto-Estimate K

21 -> 51

Fixed K: 39

Err. Threshold: 2

Boundary: Auto

Search Depth: 4096

Cite: Pulin Xie, Yongling Guo, Yue Teng, Wenbin Zhou, Yan Yu. 2024. GeneMiner: a tool for extracting phylogenetic markers from next-generation sequencing data. Molecular Ecology Resources. DOI: 10.1111/1755-0998.13924

☐ Hide Terminal Window

OK Cancel

Filter:

- Reads/File (M): Activating this option will limit the analysis to a portion of the sequencing data (M reads). When analyzing high-copy number genes, such as organellar genes, it is unnecessary to utilize the entire dataset. Engaging this option can significantly enhance the speed of the analysis. However, in cases of low sequencing depth or when dealing with low-copy number genes, it is advisable to deactivate this option.

- Filter K-value: The Filter K Value (kf) depends on three factors: acquisition rate (p) of gene-specific reads, read length (r), and variation degree (v) between reference and target sequences. Typically, with p at 0.99, r at 150, and v at 0.1, kf is around 31, our default setting. If p is 0.95 and v increases to 0.2 (r stays at 150), kf drops to 17, the proposed minimum threshold.

- Filter Step Size: Sets the interval of base numbers in the filtering process. Increasing this value can speed up the filtering process but may also reduce the recovery rate of reads. In cases of low sequencing depth or significant differences between the reference and target

sequences, it is recommended to reduce this value to 1; otherwise, keep the default setting.

For example, with a filter K-value of 7 and a step size of 1, the segmentation of reads is as follows:

sequence **ATGGAAGTCGCGGAATC**

7mers ATGGAAG
 TGGAAAGT
 GGAAGTC
 GAAGTCG
 AAGTCGC
 AGTCGCG
 GTCGCGG
 TCGCGGA
 CGCGGAA
 GCGGAAT
 CGGAATC

- High Speed: Enabling this option stores the majority of the data in memory instead of computing it in real-time. This approach can nearly double the processing speed compared to when the option is disabled. However, it is important to note that the RAM usage will also increase substantially, almost doubling in capacity.

Further Filter

- Depth Limit: When the software processes a large amount of filtering data, it initiates a secondary filtering process aimed at enhancing the efficiency and accuracy of sequence assembly. GeneMiner2 increases the filtering K value (kf) by 2 each time when the product of read length and read count divided by the average reference length exceeds this value, with an upper limit of 63.

- File Size Limit: When the software processes a large amount of filtering data, it initiates a secondary filtering process aimed at enhancing the efficiency and accuracy of sequence assembly. GeneMiner2 increases the filtering K value (kf) by 2 each time when the size of the filtering file exceeds this value (measured in MB), with an upper limit of 63.

Assemble

- [Auto Estimate K]: Sets the starting and terminal value for the automatic estimation process of the assembly K-mer (ka). Recommended check.

- [Fixed Assembly K-mer Value]: Utilize a fixed assembly K-mer (ka) value for sequence assembly.

- [Error Threshold]: Within the generated Kmers from reads, only Kmers with a count exceeding this specified value will be used for sequence assembly.

Note: A higher error threshold improves accuracy. Use the default value if few sequences are recovered.

- [Boundary]: In GeneMiner2, the soft boundary is the maximum extension outside sequence edges during target sequence recovery. 'Auto' limits this to half the read length. Setting it to '0' disables it, while 'Unlimited' imposes no boundary length restriction.

- [Search Depth]: This option specifies the maximum number of distinct candidate sequences retained during assembly, which typically requires no modification. For target sequences exceeding 5k in length, a higher value may be set.

[Analysis > Trim With Reference]:

- [Trim Terminal]: Trims and joins both ends of matched reference regions to generate high-quality assemblies. Recommended when the reference and sequencing data are from the same source with high sequencing depth.

- [All Fragments]: Uses all matched reference fragments for alignment and trimming. Suitable for low coverage datasets needing more and longer sequences. Recommended when the reference and sequencing data are from different sources.

- [Longest Fragments]: Retains only the longest matched region for each reference fragment. Suitable for genomes with paralogs or transcriptomes with alternative splicing.

[Analysis > Iteration > First Iteration]: Use the sequences in `contigs_all` from the output directory as the reference sequence, and re-execute all filtering and assembly processes. The results are saved in the `iteration` folder in the output directory. Running this can enhance the length and accuracy of the sequences.

[Analysis > Iteration > Cover with Iteration]: Replace the result files in `results` with the post-iteration result files.

[Analysis > Iteration > Multiply Iteration]: Perform multiple iterations to enhance the length and accuracy of the sequences.

[Analysis > Get Best Reference]: The reference with the highest number of matching kmers is selected as the best reference sequence.

[Analysis > Generate Consensus]: Map the result sequences to the filtered fq files. Set the threshold according to the prompts; increasing the threshold will increase the number of ambiguous bases. If you want to differentiate mixed sequences, it is recommended to choose the default (0.75), and if you want results without degenerate bases, it is recommended to choose 0.25.

[Analysis > MultiCopy Detection]: Detect multi-copy genes, which are stored in the "multicopy" file.

[Analysis > Plant Chloroplast Genome]: GeneMiner2 uses NOVOPlasty for organelle

genome assembly. The software provides downloadable chloroplast genome sequences of closely related species as reference sequences and selects a closely related one as the seed sequence, which can resolve issues with inverted repeats in chloroplast regions. After loading the data file, the chloroplast genome assembly can begin. Typically, the default parameters are sufficient. For more detailed default settings, manually edit the `NOVO_config.txt` file in the `analysis` directory of the application package. Be careful not to delete content between \$ symbols. Click the confirm button to start, and all results will be saved in the `Organelle` folder in the output directory.

NOVOPlasty

Project

Type Length

☒ Pre-filter with Reference Sequence

K-mer Read Length

Memory Insert Size

Chloroplast Sequence

Reference Sequence

NOVOPlasty - The organelle assembler and heteroplasmy caller
Cite:
Dierckxsens N., Mardulyn P. and Smits G. (2016)
NOVOPlasty: De novo assembly of organelle
genomes from whole genome data. *Nucleic
Acids Research* doi: 10.1093/nar/nkw055

[Analysis > Plant Mitochondrial Genome]: GeneMiner2 uses NOVOPlasty for organelle genome assembly. **Previously assembled chloroplast genomes must be selected**, then load the data file to begin the mitochondrial genome assembly. The software provides downloadable mitochondrial genome sequences of closely related species as reference sequences.

[Analysis > Animal Mitochondrial Genome]: GeneMiner2 uses NOVOPlasty for organelle genome assembly.

NOVOPlasty settings can be adjusted based on computer configuration to increase the allowable maximum memory for enhanced speed. For detailed explanations of the parameters, see [NOVOPlasty Github](<https://github.com/ndierckx/NOVOPlasty>).

[Analysis > Find Single Copy Genes]: Transcript data needs to be imported as the reference sequence, and single-copy genes are extracted from the imported transcripts, which will subsequently be used as references.

5.3.3 Batch

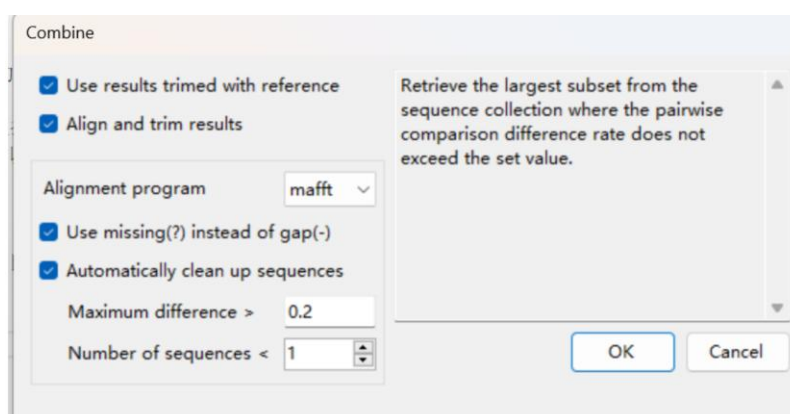
[Batch > Filter]: Use references to filter sequencing data in batch, obtaining reads associated with target genes.

[Batch > Filter & Assemble]: Perform batch analysis on selected sequencing files from different species, automatically completing all steps of filtering, (further filtering), and assembly using the current settings. All results are saved in the output directory named after the sequencing file names. **Details see in Analysis.**

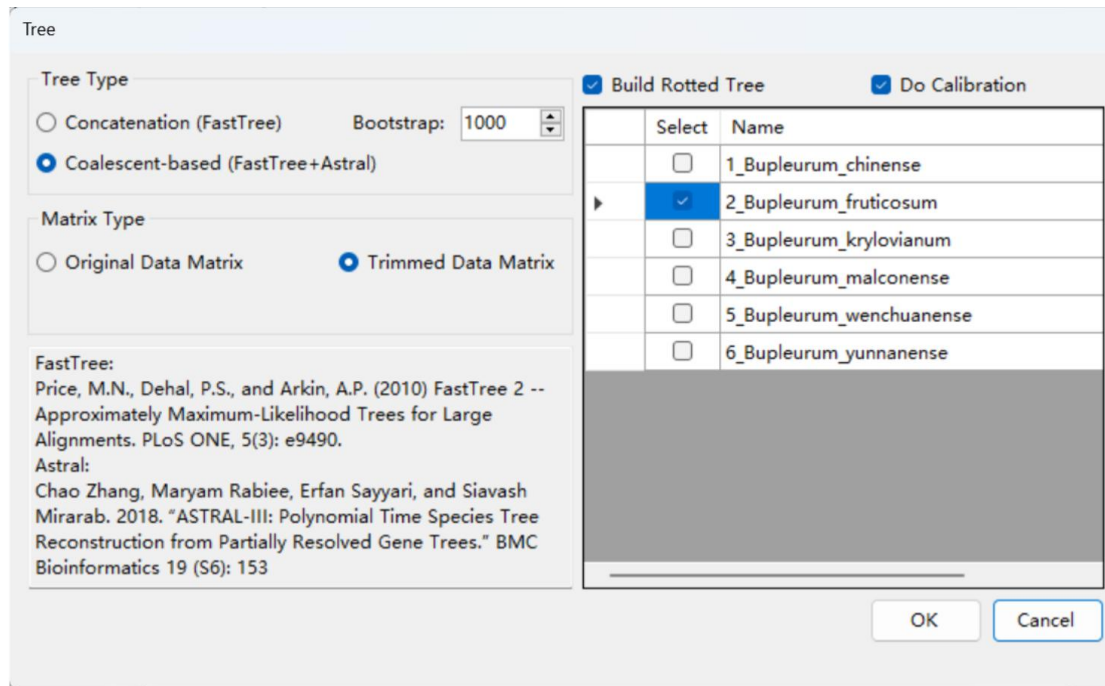
[Batch > Re-Assemble]: Perform batch analysis on selected sequencing files from different species, reassembling using the current settings.

[Batch > Trim With Reference]: Use BLAST to trim low-quality sequences at both ends based on the source of the reference sequence and filter the sequences according to set thresholds. When the sequencing and reference sequence sources differ (for example, using genomic data with a transcriptome reference), it is recommended to set this value to 0 to prevent over-filtering. When the sources are the same, the default value of 50% is recommended. The trimmed results are saved in the 'blast' folder.

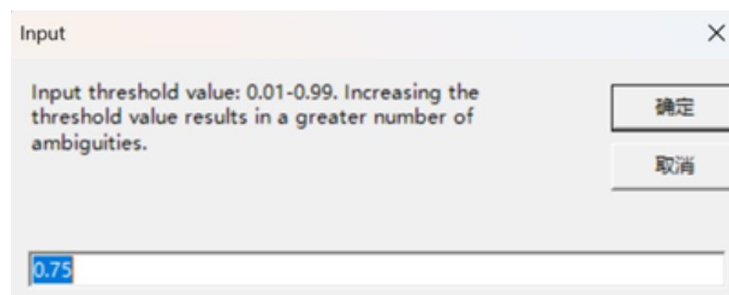
[Batch > Combine Results]: Merge the results of batch analyses and perform multiple sequence alignment and trimming on the merged fasta file.



[Batch > Build Phylogenetic Tree]: Construct a phylogenetic tree based on merged results. Concatenated/coalescent-based trees can be chosen for construction, and rooted trees can be supported with the selection of an outgroup.



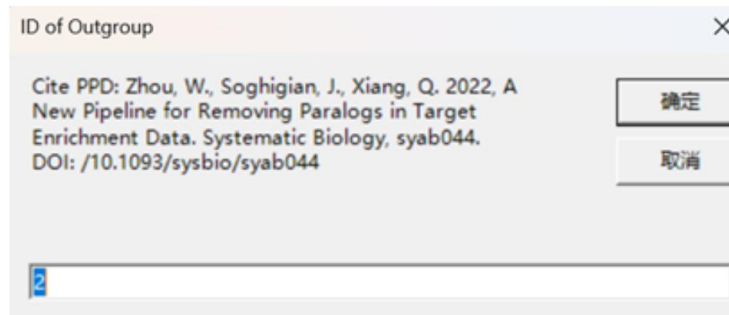
[Batch > Generate Consensus]: Map the filtered fq files of selected species in batch to the result sequences. Set the threshold according to prompts; increasing the threshold will result in a higher number of ambiguous bases.



For specific values for threshold settings, see **[Analysis Menu]**.

[Batch > Multi-Copy Detection]: Perform multi-copy gene detection on the extracted result sequences for each species.

[Batch > Paralogous Detection (PPD)]: Note, PPD can only be performed after **Consensus Reconstruction**, i.e., after paralogous parity detection on the results extracted from **[Filter & Assemble]**. An outgroup sequencing file ID number is required.



[Batch > Plant Chloroplast Genomes]: Perform batch chloroplast genome assembly on selected sequencing files from different plant species.

[Batch > Plant Mitochondrial Genomes]: Perform batch mitochondrial genome assembly on selected sequencing files from different plant species. To assemble the plant mitochondrial genome, it is necessary to already possess the chloroplast genome first.

[Batch > Animal Mitochondrial Genomes]: Perform batch mitochondrial genome assembly on selected sequencing files from different animal species. Details in **[Analysis]**.

[Batch > Summary Statistics]: Compile the results of filtering, assembly, consensus reconstruction, and paralogous parity analysis. Explanation for summary.csv can be found in [output.md].

5.3.4 Tools

[Calculate Parameters]: Use references to filter sequencing data in batch, obtaining reads associated with target genes.

[Split Filtered FQ]: Split the filtered file into paired-end reads; results are saved in the filtered folder.

[Combine reference sequences]: Merge by species or files directly.

[Add Reference to Result]: Align assembled sequences with the reference. Results are saved in the aligned or trimmed folder. You can choose to trim terminal or the entire sequence.

[Split Reference Fasta]: Select the reference sequences you wish to slice. GeneMiner2 will slice the reference into overlapping fragments to improve alignment flexibility and sensitivity. Enter the length of the input slice and the overlap length, separated by ':'. For example: 300,150 means that each reference fragment will be 300 base pairs long, and adjacent fragments will overlap by 150 base pairs. Sliced sequences will be directly imported as reference. To save them, use “File → Export Reference” to export to a specified folder.

When is slicing useful?

The sample and reference differ significantly (e.g., different species or varieties)

The reference is a transcript or contains introns

Sequencing depth is low and matched regions are sparse or fragmented

[Build Tree with Reference]:Construct a phylogenetic tree based on the imported reference sequences.

[Tree Time Calibration]:Perform divergence time calibration on a specified tree file.

[Format Tree]:Standardize species names in the imported tree.

[中文]:Change language to Chinese.

6 Algorithm Overview

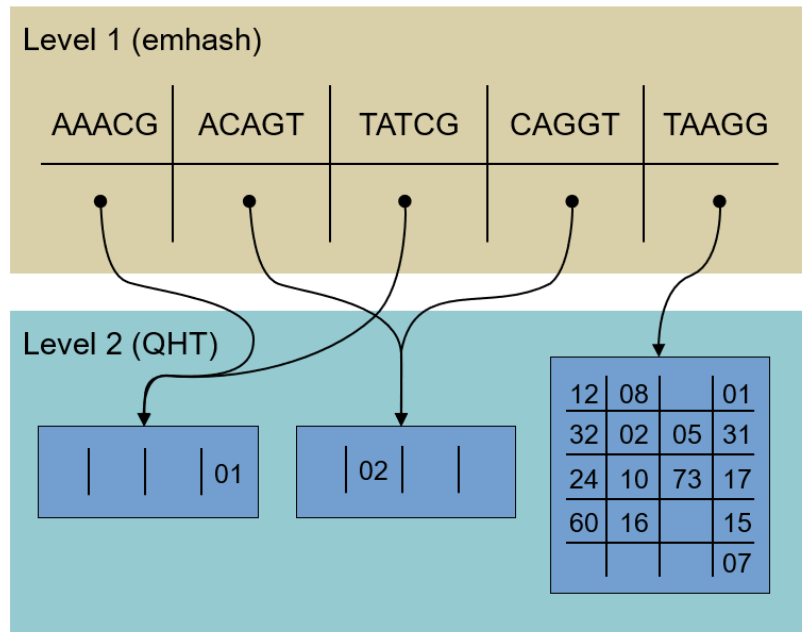
6.1 Fast read filtering

GeneMiner2 extracts a set of relevant reads from raw sequencing data before further analysis. In this step, all reference k -mers are stored into a database. Reads will be assigned to one or more genes based on their constituent k -mers. Therefore, targeted gene assembly is accelerated by operating on a reduced set of reads.

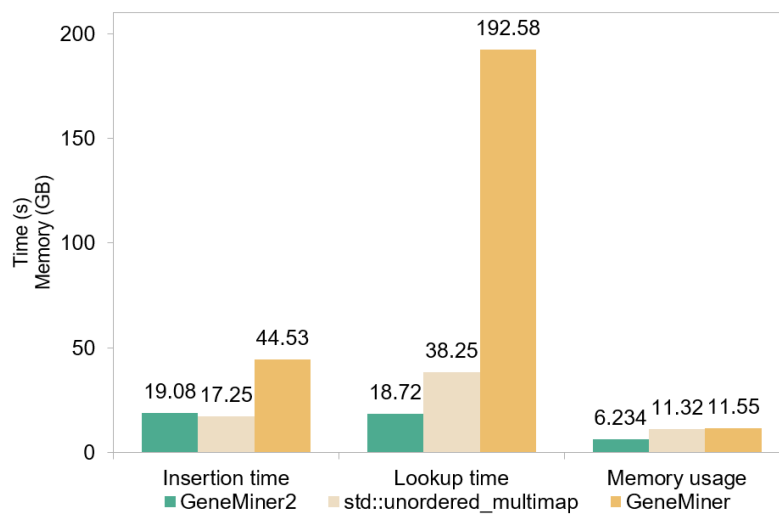
A nucleotide sequence is treated as a string over the alphabet $\{A, C, G, T\}$. Given a sufficiently long reference sequence, a reference k -mer K_i is defined as the i -th substring of length k from the reference sequence R , namely $K_i = R_j (i \leq j < i + k)$. The k -mers of a reference sequence is a sequential list $(K_1, K_2, \dots, K_{|R|-k+1})$. Similarly, the k -mers of a read is also a sequential list of all its k -substrings.

To allow efficient query with a gene's constituent k -mers, the k -mer database must be able to perform three tasks:

- (1) Given a k -mer K and a gene g , insert a key-value pair (K, g) in amortized $O(1)$ time.
- (2) Given a k -mer K and a gene g , check the existence of key-value pair (K, g) in amortized $O(1)$ time.
- (3) Given a k -mer K , find all unique key-value pairs $G = \{(K, g_1), (K, g_2), \dots\}$ in amortized $O(|G|)$ time.



Supplementary Figure 1. Illustration of the organization of the k -mer database in GeneMiner2. The Level 1 hash table (emhash) stores k -mers as keys and maps them to corresponding entries in Level 2 (QHT), which compactly stores gene indices. This structure enables fast query and iteration without subsampling, avoiding false negatives in de Bruijn assembly. The design combines efficient memory usage with high lookup speed, supporting reliable k -mer-to-gene mapping in large-scale data.



Supplementary Figure 2. The time taken and the memory consumed to insert 50 million k -

mers and look up 1 billion k -mers in each multimap implementation. Each k -mer is randomly associated with 10 out of 1,000 genes. For `std::unordered_multimap` in the C++ standard library, we used the implementation shipped with GCC 14. All benchmarks are single-threaded.

These requirements closely resemble what is known as a *multimap*. A difference is that we require key-value pair deduplication in addition to unique keys, which may not be efficient in all implementations. In the previous version of GeneMiner, the k -mer database is built around on Python’s dict type, which incurs significant overhead and limits throughput. To improve performance, GeneMiner2 implemented a custom data structure for k -mer lookup (Supplementary Fig. 1), which is shown to be at most 960% faster than the previous version (Supplementary Fig. 2).

A two-level hash table is designed as the basis of k -mer database, where Level 1 stores k -mers and Level 2 tightly packs sets of integers that represent genes. At Level 1, a generic hash table is used to map each k -mer to a pointer into its corresponding region at Level 2. Each such region at Level 2 can be interpreted as a special type of hash table, containing a set of integers. Hence, we achieve efficient value deduplication without the overhead of maintaining auxiliary data structures.

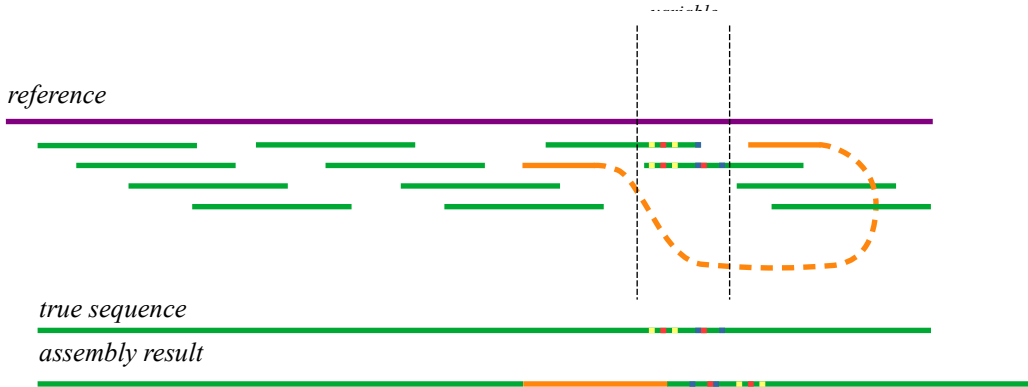
In terms of implementation, Level 1 uses *emhash* (<https://github.com/ktprime/emhash>) as the underlying storage. When inserting the relationship between a k -mer K and a gene g , we simply encode K and stored it at Level 1 with an empty value as key-value pair $(K, 0)$. We then store the presence of g in Level 2 and obtain a pointer to the corresponding region. The value in key-value pair $(K, 0)$ is then populated with the pointer. As Level 2 updates, pointers at Level 1 are continually fixed up. While slightly slowing down insertion, this allows fast k -mer queries because pointers at Level 1 always point to the correct region. In addition, *emhash* is able to reach a high load factor (> 0.9) without much performance loss, significantly reducing memory consumption.

Level 2 is implemented as a collection of Quotient Hash Tables (QHT) ^[1] with 2 buckets per row, where each QHT holds a set of integers representing all genes that contain a specific k -mer. Different from the original QHT, our implementation does not store elements as *fingerprints*, but instead achieve compactness via the ‘simple’ variant of Compact Hashing via Bucketing ^[2]. In other words, we group rows into 2^h blocks; for each gene g , we store a quotient $f(g) \bmod 2^h$ in the $\lfloor f(g)/2^h \rfloor$ -th block, where $f(x)$ is the hash function and h controls the quotient size. We also implemented partial-key cuckoo hashing, allowing an element to be stored in two rows in the same block instead of one, in a way that resembles Cuckoo Filter ^[3]. Although this change creates overhead by increasing associativity, it saves

memory by increasing load factor. Furthermore, identical QHTs inside Level 2 are deduplicated to save memory. As a result, the implementation is very memory-efficient even when written in a garbage-collected language. Overall, we enjoy amortized $O(1)$ time complexities with vastly improved memory consumption, by employing separate memory-efficient components in both levels.

QHT is designed to be probabilistic, but in the context of k -mer matching, it is reasonable to consider our variant deterministic. Because we do not use fingerprints to represent elements, false positives are eliminated. The probability of false negatives is tightly linked to the frequency of insertion failure. When a k -mer is shared by too many genes, it can be simply ignored as it provides no information in k -mer filtering, so we assume the number of genes n to be bounded in each QHT. If we allow insertion to fail at most once, we derive the failure probability $\epsilon \leq 1/n$, which results in a worst-case lower bound $l + s = \Omega(\log n / \log \log n)$ ^[4], where l is the count of buckets per row and s is the count of overflow buckets. By choosing $n = 1024$ as a generous bound, we arrive at $l = 2$ and $s = 2$, which produce no false negative on a wide range of real-world genomes. Even in the rare case of insertion failure, dropping a small number (< 10) of key-value pairs will not affect the validity of k -mer matching because a gene has hundreds of k -mers. In conclusion, our construction can provide fast, reliable k -mer matching with virtually no false negatives.

6.2 Fine-grained read selection



Supplementary Figure 3. A case where an inverted chimera causes a misassembly. The orange segment represents a read which can be mapped to the reference from two directions. The red, yellow and purple dots indicate mismatched bases between reads and reference. As the orange chimera matches the reference with fewer errors, it is assigned a higher weight than the green, structurally correct reads. Palindromic misassembly ensues when assembly progresses past the chimeric junction (dotted orange line) and continues in the reverse

direction.

While the previous version of GeneMiner has employed an iterative filtering approach to refine the reads, the removal of erroneous reads is somewhat capricious. Some types of erroneous reads significantly degrade the assembly results because they complicate the de Bruijn graph in an unexpected way. For example, multiple displacement amplification generate inverted chimeras with high abundance ^[5], causing elevated error rates in single-cell sequencing data. Due to the use of weighted de Bruijn graph, GeneMiner is especially sensitive to inversions in sequencing data: if an inversion in sequencing reads happens to skip over a variable region, it might receive a high weight due to its apparent k -mer similarity to the reference, despite having a structural difference (Supplementary Fig. 3). Additionally, even if a structural variation is resolved without error, few ways exist to leverage its phylogenetic information. Consequently, we incorporated statistical testing as well as improved heuristics into iterative read filtering to alleviate the interference of large structural variation.

Runs test When a read contains multiple inversions in relation to the reference, it becomes effectively unalignable and its location relative to the reference cannot be efficiently determined. It also tangles the de Bruijn graph by creating loops, causing unintended contig breaks. We remove such reads with a Wald–Wolfowitz runs test.

In this test, we iterate over the k -mers of the read. For each k -mer K_i , we calculate its reverse complement \bar{K}_i and test whether (K_i, g) or (\bar{K}_i, g) exists in the reference database. If (K_i, g) exists but (\bar{K}_i, g) does not, the k -mer is matching in the forward direction. Similarly, if (\bar{K}_i, g) exists but (K_i, g) does not, the match is in the reverse direction. The other two cases are ignored. The forward and reverse directionalities form a sequence, which can be tested statistically. Let the total number of forward matches be n_1 and reverse matches n_2 . We calculate the expected number of runs and the variance as follows.

$$E(R) = \frac{2n_1n_2}{n_1 + n_2} + 1$$

$$\sigma^2 = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)}$$

As most reads unambiguously match the reference in only one direction, we expect the directionality of k -mer overlaps to show strong positive autocorrelation. We require the test statistic $z = \frac{r - E(R)}{\sigma}$ to be in the lower tail with at least 10^{-5} significance to accept a read.

Although removing some reads may introduce bias and information loss, we argue that these reads contain little usable information for phylogenetics. They generate unreliable alignments and deteriorate the accuracy of phylogenetic inference. When included, they cause bias on their own rather than reduce the overall inference bias. Hence, we opt to use only reads without frequent inversions to ensure stable extraction of phylogenetic information.

Longest Run Test This test is a crude approximation of the well-known test for multiple hits on the same diagonal of a comparison matrix ^[6]. We ignore the distance between hits and inspect only the longest run in each direction, which massively speeds up the test. The null hypothesis is that matching k -mers have random directions with probability 0.5. While the random hypothesis is already rejected using the runs test, we have to note that the longest run test focuses on local k -mer chains rather than global randomness. In this regard, we do not use a conditional statistical, because this test has little equivalence to the runs test and is not a refinement.

A very accurate estimator of the longest run is given by Flajolet and Sedgewick ^[7], where n is the sample size, γ the Euler–Mascheroni constant and $P(x)$ an oscillating function with small fluctuations ($< 10^{-6}$).

$$\begin{aligned} E_n(L) &= \log_2 n + \frac{\gamma}{\ln 2} - \frac{3}{2} + P(\log_2 n) + O\left(\frac{(\ln n)^2}{\sqrt{n}}\right) \\ &\approx \log_2 n - 0.6676 \end{aligned}$$

It is known that L follows a Gumbel distribution asymptotically. Considering each run as an independent sample from a geometric distribution, we obtain a lower bound of $P(L \leq E_n(L) + m)$ by substituting it with the probability that all $E(R)$ expected runs are shorter than $E_n(L) + m$. With $m = 4$, we also show that the probability is monotonically increasing over n .

$$\begin{aligned} P(L \leq E_n(L) + 4) &\approx \left(1 - 0.5^{E_n(L)+5}\right)^{E(R)} \\ &= \left(1 - 0.5^{\log_2 n + \frac{\gamma}{\ln 2} + 3.5}\right)^{\frac{n}{2}+1} \\ &\approx \left(1 - \frac{0.0496}{n}\right)^{\frac{n}{2}+1} \end{aligned}$$

$$\begin{aligned}
& \frac{d}{dn} P(L \leq E_n(L) + 4) \\
& \approx \frac{\left(\sqrt{\frac{n - 0.0496}{n}} \right)^n \left(0.0496n + (n - 0.0496)n \ln \left(\frac{n - 0.0496}{n} \right) + 0.0992 \right)}{2n^2} \\
& > 0
\end{aligned}$$

When $n = 2$, $P(L \leq E_n(L) + 4) \approx 0.9510 > 0.95$, so the inequality holds for every $n \geq 2$. Therefore, under the null hypothesis, the longest run generally satisfies $L \leq E_n(L) + 4$ with 95% confidence. Indeed, the results by Makri and Psillakis ^[8] supports that the expected occurrences of runs whose length satisfy $L > E_n(L) + 4$ are numerically less than 0.05. This condition allows us to reject the null hypothesis with 95% significance in a one-tailed test. We also show that this test is not redundant: as $E_n(L) + 4 > \frac{n}{E(R) - 3.74\sigma}$ holds for $n \geq 19$, reads that pass the runs test do not necessarily have long k -mer runs. Consequently, we can remove reads without long same-direction k -mer runs and assign directions to the remaining reads depending on the direction of their longest run. If a read has long runs from both directions, we employ a third test to determine the validity of the read.

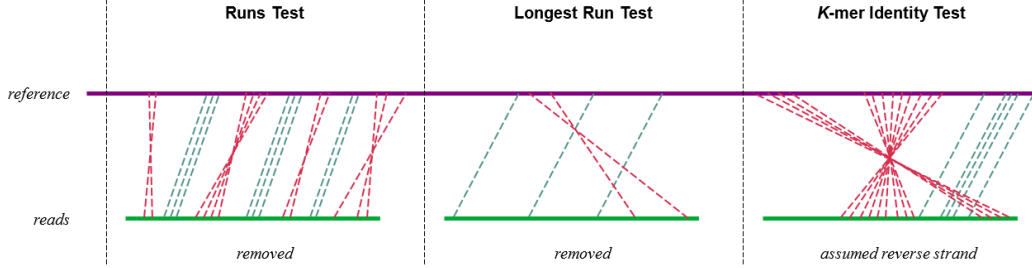
K-mer Identity Test If a read has long runs of k -mers in both directions, its orientation is considered ambiguous, but in many cases the read is perfectly valid. For example, small transposons in the reference may easily become inverted in other species, which should not be excluded from gene phylogeny. We use the percentage identity of k -mers as a heuristic to decide which region to trust. If the longest run in one direction has significantly more gaps than the other direction, it is possible that only one of the runs is actually homologous to the reference. In contrast, if two runs from opposite directions are equally similar to the reference, we have no means to resolve the ambiguity and can only remove the read.

For reads that have a forward region with n_1 k -mers in total and a reverse region with n_2 k -mers, let their proportions of k -mers matching the reference be p_1 and p_2 respectively, and we can test the null hypothesis $p_1 = p_2$ with a z -score following the standard normal distribution.

$$z = \frac{p_1 - p_2}{\sqrt{\frac{p_1(1-p_1)}{n_1} + \frac{p_2(1-p_2)}{n_2}}}$$

If the z -score does not reject the null hypothesis with 95% significance in a two-tailed test, the read is removed. Otherwise, the read is kept because it might have spanned an element with a different history from its flanks. The assembler would be able to differentiate

its orientation based on the uneven similarity to the reference. As a result, we can make use of information in most reads without risking misassembly.



Supplementary Figure 4. Illustration of the statistical tests employed in read filtering. Reads with frequent inversions are removed using the runs test. Reads without multiple consecutive k -mer hits are removed using the longest run test. Reads that seem to span an inversion are checked further; only when the hits in two directions have different similarity to the reference will the read be kept.

A summary of the three tests are given in **Supplementary Figure 4**. It should be noted that the statistical tests mentioned above require small k -mer sizes. Because most reads have no orientation conflicts, less than 1% of total reads are removed in this step. This allows us to improve the assembly of certain difficult genes without sacrificing accuracy. In addition, we implemented an improved strategy for read filtering. We remove discordant read pairs eagerly using the conclusions drawn from statistical tests. If the coverage drops below a threshold during iterative filtering, we revert the last iteration and stop filtering. This prevents overzealous read removal in low coverage regions, resulting in higher contiguity and thus better sequence recovery.

6.3 Adaptive k -mer selection

GeneMiner2 assembles individual genes by walking on the de Bruijn graph ^[9]. This approach is known to be sensitive to the selection of k -mer size ^[10]. Hence, in order to retrieve high-quality sequences, it is of paramount importance to choose the optimal k -mer size.

Anchor k -mers are defined as k -mers that occur once in a read and overlap with the reference; unique anchor k -mers refer to the first k -mers in each consecutive run of anchors.

Intuitively, the more unique anchor k -mers, the more chance to align a read correctly onto the reference. However, the possibility of misassembly also grows with the number of

unique anchor k -mers. Reducing k increases the number of anchor k -mers, but also increases coincidental k -mer overlaps. Additionally, larger values of k is generally desirable to improve assembly in repetitive regions. Unfortunately, it is not possible to obtain the best value of k analytically, because a large k -mer not only reduces spurious hits, but also helps in differentiating paralogs, whose statistical properties are largely unknown. Hence, we assign a score to k values heuristically to estimate the optimal value of k .

The strategy is to maximize k until the score falls below a threshold. Firstly, a limit on the length of consecutive anchor k -mers is set. If a run of anchor k -mers is long, more than one unique k -mers will be counted, allowing it to win over coincidental overlaps. Secondly, instead of directly using the number of unique anchor k -mers N , we define the score as

$$s = \frac{kN}{l_r - k + 1}$$

, where l_r is the length of the read. The score can be understood as normalized anchored bases: kN is the number of bases covered by unique anchor k -mers and $l_r - k + 1$ the number of k -mers in the read. As the read gets longer, the expected number of anchors is also higher, so the number of anchored bases is normalized by the total number of k -mers, reducing the bias favoring small k -mers on relatively short reads. Lastly, we find the maximum k such that s is higher than half its maximum value. This is because the weighted de Bruijn graph assembler can work reasonably well when there are few k -mer overlaps between the reads and the reference. While we do not want to lose most of the anchors, we consider it acceptable to reject half of anchored bases. Therefore, shorter anchor k -mers are sacrificed to the benefits of a larger k , which make assembly more accurate. This strategy both lowers the error rate and improves the contiguity, enabling reliable phylogenetic inference.

6.4 Parameters calculation and settings

Genome assemblers are among the most user-unfriendly pieces of software in bioinformatics. Determining the appropriate parameters is generally an arduous task because of the complex interaction between parameters ^[11].

The tool calculates a several parameters from 4 arguments: the variation between samples (v), the average length of genes (l_g), the average sequencing depth (d) and the read length (l_r). We expect these arguments to be relatively easy to understand: only v necessitates

an educated guess based on average nucleotide identity. The other three arguments can be inferred from the properties of the samples, easing the process of software configuration.

The k -mer size in read filtering (k_f) is calculated based on the read acquisition rate $p = 1 - (1 - (1 - v)^{k_f})^{l_r - k_f + 1}$ using the formula proposed in the previous version of GeneMiner^[12].

The tool tries twice to find the largest k_f : first using $p \geq 0.99$ as the constraint; $p \geq 0.95$ if the former is not possible. The lower limit is $k_f \geq 17$ as per the original recommendation.

Otherwise, an error will be generated.

6.5 Trimming Strategies

When sequencing depth was high and reference and sequencing data originated from the same source, the “*Trim Terminal*” mode was applied to excise and concatenate the terminal regions of the reference sequence. When sequencing depth was lower or reference and sequencing sources differed, the “*All Fragments*” mode was used to retain all alignable regions for complete sequence generation. For loci involving paralogous genes or alternative splicing, the system activated the “*Longest Isoform*” mode to maximize the retained fragment length.

7 FAQ

1. Why are there no assembled results?

- The selected reference may be too distantly related. Try a closer reference.
- The sequencing depth may be insufficient. Consider lowering the "Filter K " value or increasing the "Reads/File(M)" parameter.
- If the results are not satisfactory, rerun [Analysis > Iteration] to improve contigs.
- You can also reduce the "Assembly K mer" value. **Note: Lower values may reduce accuracy.**
- The recommended minimum K mer is >17 and should be an odd number.

3. What are the memory requirements?

GeneMiner2 is optimized for low-memory usage. Adjust the number of threads based on your

system memory. The default is 7.

4. How do I extract intron sequences?

Load a GenBank file under [File > Load Reference] and confirm the prompt to import it as a gene list. You can set [Extend Left] and [Extend Right] to include intronic flanks. If a CDS contains internal introns (i.e., exon gaps), these will also be retained.

5. Why does [Batch > Combine & Trim] produce no results?

Ensure all of the following are satisfied:

- The sequencing files and reference sequences are loaded.
- A result folder exists.
- The ID of each result folder matches the ID of the corresponding input.
- Avoid using folders with Chinese characters.

6. Why is there no output for PPD (Paralogous Detection)?

PPD requires:

- A minimum of three species.
- No Chinese characters in any file path.

7. Why are mitochondrial/chloroplast genome assemblies empty?

Uncheck the [Reads/File(M)] box at [Analysis > Filter] to use the full sequencing file.

Memory availability can determine whether the assembly succeeds, insufficient or excessive memory may cause assembly failure.

If your computer is low on RAM, **be sure to adjust the memory to an appropriate level.**

Changelog

- 2025/1/24 Optimized overlap calculation in assembler.
- 2025/1/30 Replaced Python-based filter with a much faster native version (described in Algorithm Overview).
- 2025/1/31 Implemented a parallel task scheduler to improve parallelism on Windows.
- 2025/3/5 Implemented a new k-mer size estimation algorithm in assembler (described in Algorithm Overview).
- 2025/3/6 Implemented parameter estimation (Calculate Parameters) feature.
- 2025/6/10 Implemented a user-friendly command-line interface and its build system for Unix-like systems.
- 2025/6/27 Replaced maximum subset selection in Combine Results. The old code is unfortunately dependent on the ordering of sequences and attempts to solve an NP-complete problem. The new code clusters sequences and finds biconnected components (which give at least some phylogenetic information).

References

- [1] SIMON 3 E B I B E G N K A M E R A G S G S A U - V A W, 5 R G I B I A J F G R P G, 6 B A P, et al. Initial sequencing and comparative analysis of the mouse genome [J]. 2002, 420(6915): 520-62.
 - [2] RáDAI Z, VáRADI A, TAKÁCS P, et al. An overlooked phenomenon: complex interactions of potential error sources on the quality of bacterial de novo genome assemblies [J]. 2024, 25(1): 45.
 - [3] XIE P, GUO Y, TENG Y, et al. GeneMiner: A tool for extracting phylogenetic markers from next-generation sequencing data [J]. Mol Ecol Resour, 2024, 24(3): e13924.
 - [4] GÉRAUD R, LOMBARD-PLATET M, NACCACHE D. Quotient hash tables: efficiently detecting duplicates in streaming data; proceedings of the Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, F, 2019 [C].
 - [5] KÖPPL D, PUGLISI S J, RAMAN R J A. Fast and simple compact hashing via bucketing [J]. 2022, 84(9): 2735-66.
 - [6] FAN B, ANDERSEN D G, KAMINSKY M, et al. Cuckoo filter: Practically better than bloom; proceedings of the Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, F, 2014 [C].
 - [7] YEO K. Cuckoo hashing in cryptography: Optimal parameters, robustness and applications; proceedings of the Annual International Cryptology Conference, F, 2023 [C]. Springer.
 - [8] LU N, QIAO Y, LU Z, et al. Chimera: The spoiler in multiple displacement amplification [J]. 2023, 21: 1688-96.
 - [9] WILBUR W J, LIPMAN D J J P O T N A O S. Rapid similarity searches of nucleic acid and protein data banks [J]. 1983, 80(3): 726-30.
 - [10] FLAJOLET P, SEDGEWICK R. Analytic combinatorics [M]. cambridge University press, 2009.
 - [11] MAKRI F S, PSILLAKIS Z M J C, APPLICATIONS M W. On success runs of a fixed length in Bernoulli sequences: Exact and asymptotic results [J]. 2011, 61(4): 761-72.
 - [12] IDURY R M, WATERMAN M S J J O C B. A new algorithm for DNA sequence assembly [J]. 1995, 2(2): 291-306.
 - [13] SCHATZ M C, DELCHER A L, SALZBERG S L J G R. Assembly of large genomes using second-generation sequencing [J]. 2010, 20(9): 1165-73.
 - [14] CHIKHI R, MEDVEDEV P J B. Informed and automated k-mer size selection for genome assembly [J]. 2014, 30(1): 31-7.
 - [15] SHARIAT B, MOVAHEDI N S, CHITSAZ H, et al. HyDA-Vista: towards optimal guided selection of k-mer size for sequence assembly [J]. 2014, 15: 1-8.
-
- [1] GÉRAUD R, LOMBARD-PLATET M, NACCACHE D. Quotient hash tables: efficiently detecting duplicates in streaming data; proceedings of the Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, F, 2019 [C].
 - [2] KÖPPL D, PUGLISI S J, RAMAN R J A. Fast and simple compact hashing via bucketing [J]. 2022, 84(9): 2735-66.
 - [3] FAN B, ANDERSEN D G, KAMINSKY M, et al. Cuckoo filter: Practically better than bloom; proceedings of the Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, F, 2014 [C].
 - [4] YEO K. Cuckoo hashing in cryptography: Optimal parameters, robustness and applications; proceedings of the Annual International Cryptology Conference, F, 2023 [C]. Springer.
 - [5] LU N, QIAO Y, LU Z, et al. Chimera: The spoiler in multiple displacement amplification [J]. 2023, 21: 1688-96.
 - [6] WILBUR W J, LIPMAN D J J P O T N A O S. Rapid similarity searches of nucleic acid and protein data banks [J]. 1983, 80(3): 726-30.

- [7] FLAJOLET P, SEDGEWICK R. Analytic combinatorics [M]. cambridge University press, 2009.
- [8] MAKRI F S, PSILLAKIS Z M J C, APPLICATIONS M W. On success runs of a fixed length in Bernoulli sequences: Exact and asymptotic results [J]. 2011, 61(4): 761-72.
- [9] IDURY R M, WATERMAN M S J J O C B. A new algorithm for DNA sequence assembly [J]. 1995, 2(2): 291-306.
- [10] SCHATZ M C, DELCHER A L, SALZBERG S L J G R. Assembly of large genomes using second-generation sequencing [J]. 2010, 20(9): 1165-73.
- [11] RÁDAI Z, VÁRADI A, TAKÁCS P, et al. An overlooked phenomenon: complex interactions of potential error sources on the quality of bacterial de novo genome assemblies [J]. 2024, 25(1): 45.
- [12] XIE P, GUO Y, TENG Y, et al. GeneMiner: A tool for extracting phylogenetic markers from next-generation sequencing data [J]. Mol Ecol Resour, 2024, 24(3): e13924.