CyDotian's manual

An algorithm for identifying internal repeats of nucleic acid and amino acid sequences.

Authors	Huilong Chen
Email	chenhuilong131@163.com

Overview

CyDotian algorithm is a dynamic programming algorithm, which can identify all internal repeats of the sequence itself that allow mutation. In order to achieve efficient output of the results, we used C, the fastest underlying computer language available, to implement the algorithm and allow it to be compiled into an executable program. Downstream analysis tools are then written in Python, the most popular language for data processing and visualisation. These downstream analysis tools include processing the location and number of repeat segments, plotting dotplots, plotting depth plots, calculating repetition density and outputting specific repeat segment comparison details. All are batch processed and exported, which is extremely user-friendly. Users can suggest and optimise the development of all codes. Moreover, due to the applicability of the CyDotian algorithm, it can also be used to identify similarity segments between two different sequences that allow for mutations. In addition, by replacing the U in the RNA sequence with a T, CyDotian can be used to find all the reverse complementary sequences, helping to predict the stem-loop structure of the RNA molecule.

Usage information is built into the program. To display usage on the screen, the user simply runs the program by specifying the -h/--help parameter:

$ python3 program_name.py -h/--help (for Python scripts)

The following is the list of executable programs:

bpRepeatScan (used for nucleic acid sequences)

aaRepeatScan (used for amino acid sequences)

slidingWindow (used for used for nucleic acid and amino acid sequences via sliding window method)

Parameter configuration file

CyDotian.config (used for CyDotian algorithm)

CyDotian_exact_match.config (used for MUMmer's repeat-match algorithm)

CyDotian_sliding_window.config (used for sliding window algorithm)

Batch processing programs:

Tool 1. 1.0_batch_check_sequence.py

Tool 2. 1.1_batch_run_CyDotian.py

Tool 3. 1.2_batch_run_draw_dotplot.py

Tool 4. 1.3_batch_run_draw_depth_plot.py

Tool 5. 1.4_batch_run_output_repeat_density.py

Tool 6. 1.5_batch_extract_repeat_sequences.py

Tool 7. 1.6_Extract_the_corresponding_results_by_name.py

Tool 8. 1.7_batch_run_CyDotian_in_pairwise_comparison_mode.py

Tool 9. 1.8_batch_extract_repeat_sequences_in_pairwise_comparison_mode.py

Tool 10. 2.1_batch_run_CyDotian_exact_match.py

Tool 11. 2.7_batch_run_CyDotian_exact_match_in_pairwise_comparison_mode.py

Tool 12. 3.1_batch_run_CyDotian_sliding_window.py

Tool 13. 3.2_batch_run_CyDotian_sliding_window_in_pairwise_comparison_mode.py

Tool 14. 3.3_batch_run_draw_dotplot_sliding_window.py

Articles that have cited the CyDotian algorithm:

Ge W, Chen H, Zhang Y, et al. Integrative genomics analysis of the ever-shrinking pectin methylesterase (PME) gene family in foxtail millet (Setaria italica)[J]. Functional Plant Biology, 2022, 49(10): 874-886.. https://doi.org/10.1071/FP21319

Documentation

The online documentation is located at the GitHub Wiki.

Dependencies

GCC and Python3

The CyDotian toolkit code can be easily ported to Windows and Mac OS systems with a few simple modifications. However, we recommend that the CyDotian toolkit should be run on Linux/Unix servers due to the amount of memory available.

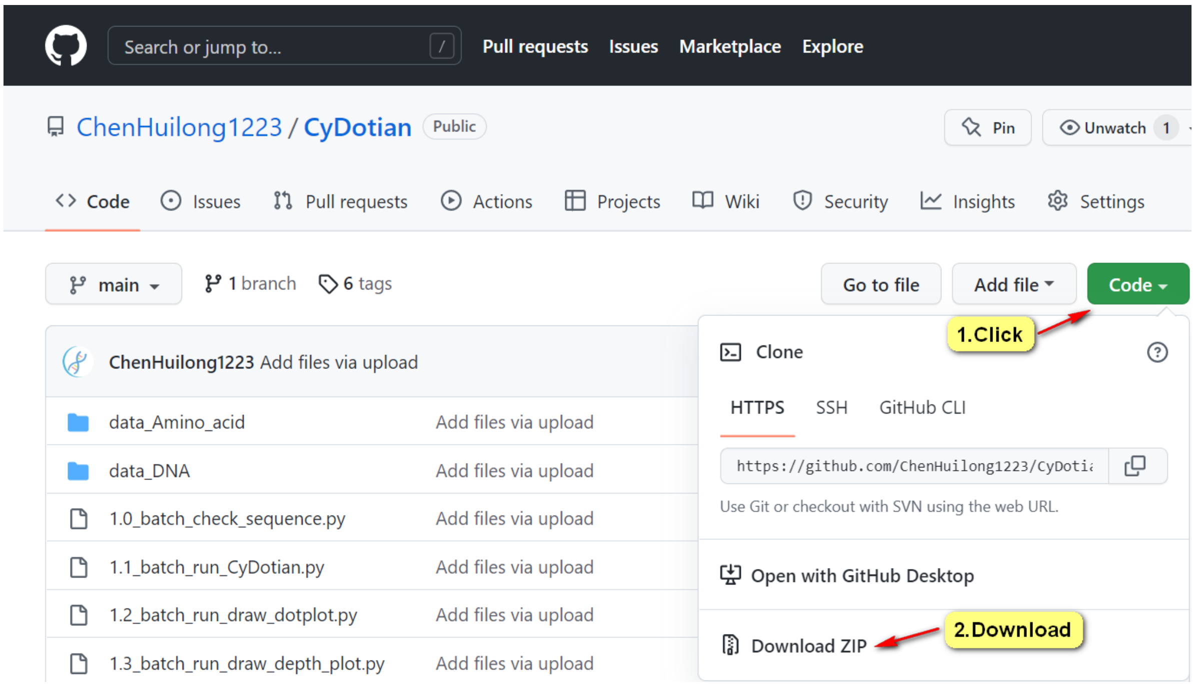
Installation

Open a terminal and follow the steps below to enter the command.

```
$ mkdir ~/CyDotian # or any directory of your choice.

$ cd ~/CyDotian
```

Manually download the "CyDotian" package



ChenHuilong1223 Add files via upload

data_Amino_acid Add files via upload
data_DNA Add files via upload
1.0_batch_check_sequence.py Add files via upload
1.1_batch_run_CyDotian.py Add files via upload
1.2_batch_run_draw_dotplot.py Add files via upload
1.3_batch_run_draw_depth_plot.py Add files via upload

https://github.com/ChenHuilong1223/CyDoti

Use Git or checkout with SVN using the web URL.

Put "CyDotian.zip" into the newly created folder "CyDotian". Then enter the following command.

```
$ chmod 775 CyDotian-main.zip

$ unzip CyDotian-main.zip

$ chmod -R 775 CyDotian-main

$ cd CyDotian-main

$ make
```

Or, if your server is available, you can use the following command. Instead of downloading it manually.

```
$ git clone https://github.com/ChenHuilong1223/CyDotian.git

$ cd CyDotian

$ make
```

If you see the following result output in the terminal, the installation is successful.

gcc -g -std=c99 bpRepeatScan.c -o bpRepeatScan

gcc -g -std=c99 aaRepeatScan.c -o aaRepeatScan

gcc -g -std=c99 slidingWindow.c -o slidingWindow

Tutorial

Preparation

File format: fasta format

```
>name1

sequence1 # Sequences can be displayed on multiple lines or on one line.

>name2

sequence2

...
```

If you want to analyse nucleic acid sequences, create a folder for nucleic acid sequences in the directory where the program is now located. If you want to analyse amino acid sequences, create a folder for amino acid sequences in the directory where the program is located.

We recommend that you include the word nucleic acid or amino acid in the name of the folder to make it easier to distinguish between sequence types. This is because the CyDotian toolkit does not allow batch analysis with both nucleic acid and amino acid sequences.

In the CyDotian toolkit, all file or folder paths can be either absolute or relative, but cannot have the character "/" at the end of the path. For the output folder path, there is no need to create it in advance, CyDotian will create it automatically if the destination folder path does not exist.

Here we use the analysis of DNA sequences as an example:

1. **Create a folder in the "CyDotian-main" directory**

   ```
   $ cd ~/CyDotian/CyDotian-main/

   $ mkdir test_DNA
   ```

2. **Place the DNA sequences to be analysed (in fasta format) into the created folder**

   User Notice:

   The "test_DNA" folder can contain multiple fasta files, and a fasta file can contain multiple sequences.

3. **Set the parameters as required via the "CyDotian.config" file**

   User Notice:

   When the sequence is a DNA, the valid parameters are DNA_Matrix, mode, identityThr, and repeatLen.

   When the sequence is an amino acid, the valid parameters are aminoAcidMatrix, mode, similarityThr, and repeatLen.

Parameters	Descriptions
fileType=0	#0 for DNA, 1 for amino acid.
DNA_Matrix=0	#0 for BLAST matrix, 1 for Transition-transversion matrix - only valid for DNA.

Parameters	Descriptions
aminoAcidMatrix=1	#0 for BLOSUM45, 1 for BLOSUM62, 2 for BLOSUM80, 3 for BLOSUM90, 4 for PAM30, 5 for PAM70, 6 for PAM250 - only valid for amino acid.
mode=0,1	#0 for direct repeat, 1 for inverted repeat, #2 for reverse complement - only valid for DNA, suitable for inferring reverse complement of RNA.
identityThr=0.85	#Identity threshold, indicating that only results greater than or equal to this threshold are output - only valid in the case of DNA.
similarityThr=0.85	#Similarity threshold, indicating that only results greater than or equal to this threshold are output - only valid in the case of amino acid.
repeatLen=6	#A threshold for repeat or similar fragment length, indicating that only results greater than or equal to this threshold will be output.

Start

1. Check sequences for illegal characters

- Command

```
$ python3 1.0_batch_check_sequence.py -f 0 -s test_DNA
```

Parameters	Descriptions
-f	Input sequence type, 0 for DNA, 1 for amino acid
-s	Input the path to the sequence folder
-h	Show this help message

If you see the following message in the terminal, the sequence can be analysed by CyDotian.

```
Congratulations, your file '*.fasta' can be analyzed by CyDotian!

Congratulations, your file '*.fasta' can be analyzed by CyDotian!

...

Congratulations, the script worked and finished successfully!

The program running time: *hour(s) *minute(s) *second(s).
```

2. Batch run CyDotian

- Command

```
$ python3 1.1_batch_run_CyDotian.py -c CyDotian.config -s test_DNA -o
test_DNA_result
```

Parameters	Descriptions
-c	Input the path to the "CyDotian.config" file
-s	Input the path to the sequence folder
-o	Output folder path
-h	Show this help message

If you see an output message like this, it means that your sequence has been successfully analysed. Each fasta format file corresponds to one paragraph of the following output message.

```
...

For 'positions',
This fasta file with * sequences has a total of * failures and * successes this time!
If a successful ID appears in 'failure_sequences.txt', the number result after deleting the position file corresponding to this ID is:
 This fasta file with * sequences has a total of * failures and * successes this time!
For 'positions_original',
This fasta file with * sequences has a total of * failures and * successes this time!
If a successful ID appears in 'failure_sequences.txt', the number result after deleting the position file corresponding to this ID is:
 This fasta file with * sequences has a total of * failures and * successes this time!
For 'positions' and 'positions_original', the sum of this number of successes and failures is correct!
For 'positions' and 'positions_original', the sum of this number of successes and failures and the ID is correct!

...

Congratulations, the script worked and finished successfully!
The program running time: *hour(s) *minute(s) *second(s).
```

User Notice:

For a fasta format file,

The files with the duplicate results removed are placed in the "positions" folder and the original result files are placed in the "positions_original" folder. That is, the internal repeat position files in the "positons_original" directory can be used to draw symmetrical dotplots, depth maps and calculate repetition densities.

the names and lengths of the failed sequences are recorded in the "failed_sequences.txt" file in the "positions" folder.

The reason for the failure of each sequence analysis is logged in the "failure_sequences_error_log.txt" file in the "positions" folder.

The parameter information and the description of the result file are logged in a "log.txt" file in the "positions" folder.

The names of sequences containing repeats greater than the repeatLen threshold length are recorded in the "sequence_names_*.txt" file in the "positions" folder.

The names and lengths of all sequences in the fasta format file are recorded in the "all_sequences_length.txt" file in the "positions_original" folder.

The position result files are available in two types, direct and inverted repeat.

If the sequence is DNA, the result file can include the reverse complement results.

The file "name_positions_direct.txt" is the result of the direct repeat of the name sequence.

The file "name_positions_inverted.txt" is the result of the inverted repeat of name sequence.

The file "name_positions_reverse_complement.txt" is the result of the reverse complement of the namesequence.

- Description of position results

  Start1: the starting position of the vertical sequence in the dotplot.

  End1: the end position of the vertical sequence in the dotplot.

  Start2: the starting position of the horizontal sequence in the dotplot.

  End2: the end position of the horizontal sequence in the dotplot.

  Length: the length of the repeat fragment.

  Identity: identity.

  Mismatch: number of mismatches.

  Similarity: similarity, which appears only when the input sequence is an amino acid.

  Score: comparison score.

Start1	End1	Start2	End2	Length	Identity	Mismatch	Similarity	Score
146	342	348	544	197	0.827411	34	0.934010	862
140	151	502	513	12	0.250000	9	0.916667	18
140	151	300	311	12	0.250000	9	0.916667	18
20	29	98	107	10	0.400000	6	0.900000	23
76	85	84	93	10	0.500000	5	0.900000	30
...								

Note: the result file is tab-separated and output in descending order of length.

3. Batch plotting of dotplots

- Command

```
$ python3 1.2_batch_run_draw_dotplot.py -l 10 -r test_DNA_result
```

Parameters	Descriptions
-l	Input a repeat length threshold, i.e. the minimum repeat length you want to draw
-r	Input the path to the results folder generated from step 2
-h	Show this help message

If you see the following message in your terminal, your program has finished running correctly.

```
Congratulations, the script worked and finished successfully!
The program running time: *hour(s) *minute(s) *second(s).
```

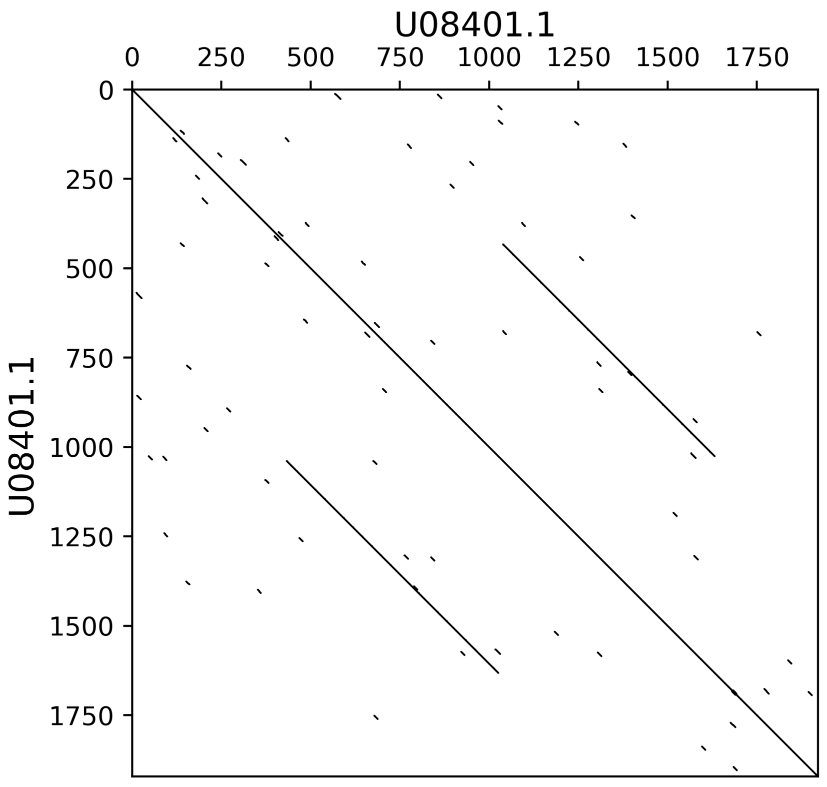
User Notice:

The script will automatically create a folder called "positions_original_dotplots" in the directory where the "positions_original" folder is located and store all the results in this folder.

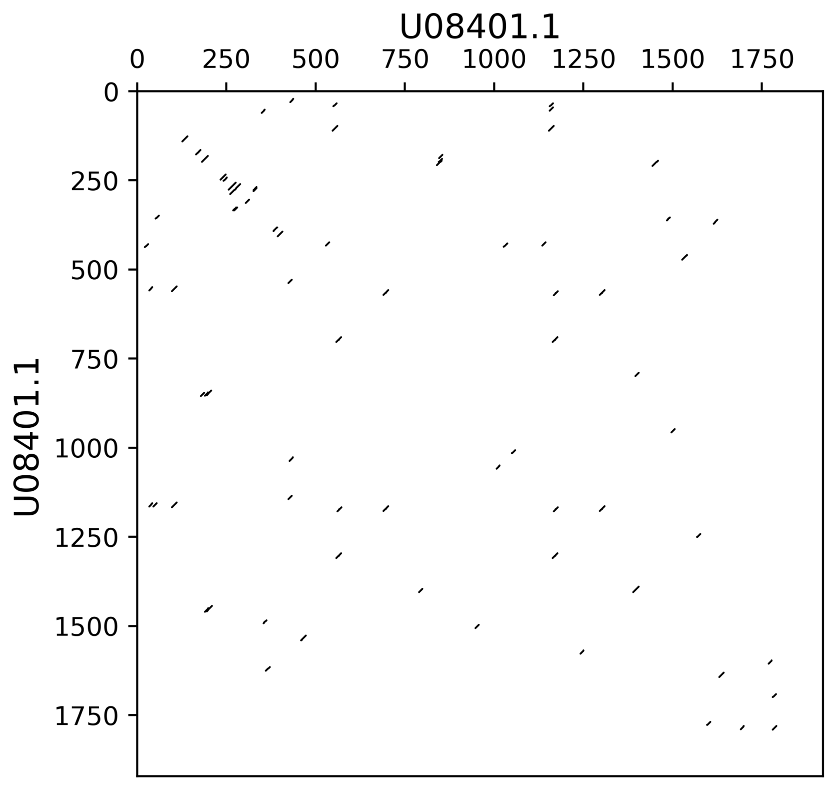
The file name of the location where the drawing failed is reiscorded in the "failure_files.txt" file in the "positions_original_dotplots" folder.

The parameter information is logged in a "log.txt" file in the "positions_original_dotplots" folder.

The file "name_direct.pdf/png" is a dotplot of the direct repeat results of the name sequence, as follows.



The file "name_inverted.pdf/png" is a dotplot of the inverted repeat results of the name sequence, as follows.



4. Batch plotting of depth maps

- Command

```
$ python3 1.3_batch_run_draw_depth_plot.py -l 10 -r test_DNA_result
```

Parameters	Descriptions
-l	Input a repeat length threshold, i.e. the minimum repeat length you want to analyze
-r	Input the path to the results folder generated from step 2
-h	Show this help message

If you see the following message in your terminal, your program has finished running correctly.

```
Congratulations, the script worked and finished successfully!
The program running time: *hour(s) *minute(s) *second(s).
```

User Notice:

The script will automatically create a folder called "positions_original_depths" in the directory where the "positions_original" folder is located and store all the results in this folder.

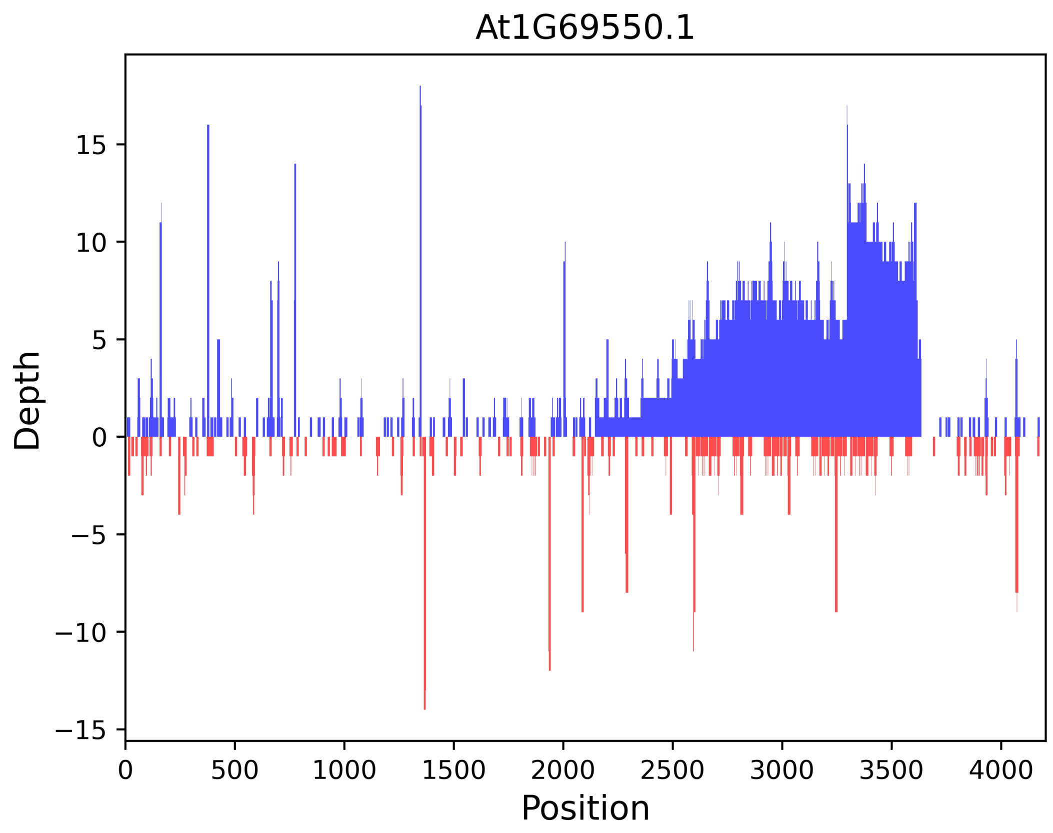
The file name of the location where the drawing failed is reiscorded in the "failure_files.txt" file in the "positions_original_depths" folder.

The parameter information is logged in a "log.txt" file in the "positions_original_depths" folder.

The "name_direct_depths.txt" file is a count of the depths of the direct repeat results dotplots.

The "name_inverted_depths.txt" file is a count of the depths of the inverted repeat results dotplots.

The "name_depth.pdf/png" file is a depth map of the internal repeat dotplots of the name sequence, with the bars above the 0 scale (blue) being the direct repeat depth map and the bars below the 0 scale (red) being the inverted repeat depth map, as follows.



5. Batch calculation of repetitive density

- Command

```
$ python3 1.4_batch_run_output_repeat_density.py -l 10 -r test_DNA_result
```

Parameters	Descriptions
-l	Input a repeat length threshold, i.e. the minimum repeat length you want to analyze
-r	Input the path to the results folder generated from step 2
-h	Show this help message

If you see the following message in your terminal, your program has finished running correctly.

```
Congratulations, the script worked and finished successfully!
The program running time: *hour(s) *minute(s) *second(s).
```

User Notice:

The script will automatically create a folder called "positions_original_densities" in the directory where the "positions_original" folder is located and store all the results in this folder.

The names of the failed sequences are recorded in the "failure_files.txt" file in the "positions_original_densities" folder.

The parameter information is logged in a "log.txt" file in the "positions_original_densities" folder.

For a fasta format file,

The direct repetition density of all sequences in this file is recorded in the "fasta format file name_direct_densities" file, as follows.

Name	Direct repetition density
U08401.1	0.000019
U08403.1	0.000016

The inverted repetition density of all sequences in this file is recorded in the "fasta format file name_inverted_densities" file, as follows.

Name	Inverted repetition density
U08401.1	0.000020
U08403.1	0.000016

Total repetition density of all sequences in this file is recorded in the "fasta format file name_total_densities" file, as follows.

Name	Total repetition density
U08401.1	0.000039
U08403.1	0.000032

Note: Slice a sequence into the different sequence regions that the user wants to compare, store them as different fasta format files, run CyDotian again following the process above, and finally plot the density result file as a graph that can be used to compare the repetition of different regions of the same sequence by the repetition density index. See, for example, this literature.

6. Batch extraction of repetitive sequences

- Command

```
$ python3 1.5_batch_extract_repeat_sequences.py -l 10 -r test_DNA_result -s
test_DNA
```

Parameters	Descriptions
-l	Input a repeat length threshold, i.e. the minimum repeat length you want to analyze
-r	Input the path to the results folder generated from step 2
-s	Input the path to the sequence folder from step 1
-h	Show this help message

If you see the following message in your terminal, your program has finished running correctly.

Congratulations, the script worked and finished successfully!
The program running time: *hour(s) *minute(s) *second(s).

User Notice:

The script will automatically create a folder called "positions_repeat_sequences" in the directory where the "positions" folder is located and store all the results in this folder.

The names of the failed sequences are recorded in the "failure_files.txt" file in the "positions_repeat_sequences" folder.

The parameter information is logged in a "log.txt" file in the "positions_repeat_sequences" folder.

The "name_direct_repeat_sequences.txt" file is the result of a direct repeat sequence within a name sequence, as follows.

Start1	End1	Start2	End2	Length	Identity	Mismatch	Score
652	664	679	691	13	0.923077	1	56
vertical							
ATCAAGTACGCCG	#Vertical	sequence	in	dotplot			
ATCGAGTACGCCG	#horizontal	sequence	in	dotplot			
horizontal							
675	684	1038	1047	10	1.000000	0	50
vertical							
CGCCATCGAG							
CGCCATCGAG							
horizontal							
...							

Note: The numbers in this file are interpreted as in the position file in step 2, and if the sequence type is amino acid, then there will be a column of Similarity results between Mismatch and Score.

The "name_inverted_repeat_sequences.txt" file is the result of a inverted repeat sequence within a name sequence. The results are interpreted as in the "name_direct_repeat_sequences.txt" file.

The "name_reverse_complement_sequences.txt" file is the result of a reverse complement sequence within a name sequence. This is only possible if the sequences analysed are of the DNA type. The results are interpreted as in the "name_direct_repeat_sequences.txt" file.

7. Extract all results for a subset of the sequence

First, prepare a subset names file for the target fasta format file, for example, the following example "subset_names.txt" file.

```
name1
name2
...
```

- Command

```
$ python3 1.6_Extract_the_corresponding_results_by_name.py -n subset_names.txt -r ./test_DNA_result/CyDotian_example_DNA.fasta -o CyDotian_example_DNA.fasta_result
```

Parameters	Descriptions
-n	Input the path to a subset file of sequence names for a target fasta format file
-r	Input the path to the folder with the same name as the corresponding fasta format file in the results folder generated in step 2
-o	Output folder path
-h	Show this help message

If you see the following message in your terminal, your program has finished running correctly.

> Congratulations, the script worked and finished successfully!
> The program running time: *hour(s) *minute(s) *second(s).

User Notice:

> The script will automatically create the folder directory specified after the -o parameter.
>
> Be sure to provide the corresponding subset name file and the target fasta format file.

8. Batch comparison of sequences in two fasta format files

- Command

```
$ python3 1.7_batch_run_CyDotian_in_pairwise_comparison_mode.py -c
CyDotian.config -v ./test_DNA/CyDotian_example_DNA.fasta -l
./test_DNA/Ath_example_DNA2.fasta -o pairwise_comparison_result
```

Parameters	Descriptions
-c	Input the path to the "CyDotian.config" file
-v	Input the path to the fasta format file in the vertical direction in the dotplot
-l	Input the path to the fasta format file in the horizontal direction in the dotplot
-o	Output folder path
-h	Show this help message

If you see the following message in your terminal, your program has finished running correctly.

> ...
>
> Congratulations, the script worked and finished successfully!
> The program running time: *hour(s) *minute(s) *second(s).

User Notice:

> The script will automatically create the folder directory specified after the -o parameter.
>
> Be sure to provide the corresponding subset name file and the target fasta format file.

A total of two folders named "positions_original" and "positions_original_dotplots" will be created in the results folder. All position results will be placed in the "positions_original" folder and all dotplot results will be placed in the "positions_original_dotplots" folder.

The names of the two compared sequences are concatenated into one result name using "VS", the left one representing the vertical sequence name and the right one representing the horizontal sequence name.

All result files are interpreted as above.

9. Batch extraction of similar sequences of two sequences

- Command

```
$ python3 1.8_batch_extract_repeat_sequences_in_pairwise_comparison_mode.py -l
10 -r pairwise_comparison_result -s1 ./test_DNA/CyDotian_example_DNA.fasta -s2
./test_DNA/Ath_example_DNA2.fasta
```

Parameters	Descriptions
-l	Input a similar sequence length threshold, i.e. the minimum similar sequence length you want to analyze
-r	Input the path to the results folder generated from step 8
-s1	Enter the path to the fasta format file from step 8 in the vertical direction of the dot plot
-s2	Enter the path to the fasta format file from step 8 in the horizontal direction of the dot plot
-h	Show this help message

If you see the following message in your terminal, your program has finished running correctly.

Congratulations, the script worked and finished successfully!
The program running time: *hour(s) *minute(s) *second(s).

User Notice:

The script will automatically create a folder called "positions_original_similar_sequences" in the directory "pairwise_comparison_result".

Be sure to provide the corresponding subset name file and the target fasta format file.

The names of the two compared sequences are concatenated into one result name using "VS", the left one representing the vertical sequence name and the right one representing the horizontal sequence name.

All result files are interpreted as above.

10. Batch execution consistent with the results of MUMmer's repeat-match

To achieve exactly the same results as MUMer's repeat-match and batch-processing them, the steps are the same as those for running the CyDotian algorithm in batch.

You simply need to use “CyDotian_exact_match.config” instead of “CyDotian.config” and set the parameters in that configuration file. Since MUMmer's repeat-match algorithm only recognizes perfect repeats, the key parameter is repeatLen.

And replace the script "1.1_batch_run_CyDotian.py" with "2.1_batch_run_CyDotian_exact_match.py".

For batch analysis of comparisons between two different sequences, replace “1.7_batch_run_CyDotian_in_pairwise_comparison_mode.py” with “2.7_batch_run_CyDotian_exact_match_in_pairwise_comparison_mode.py”.

11. Batch processing of biological sequences via sliding window approach

To realize the batch for sequence comparison through the sliding window principle, the operation steps are the same as the batch run CyDotian algorithm steps.

You simply need to use "CyDotian_sliding_window.config" instead of "CyDotian.config" and set the parameters in that configuration file. The key parameters are ideSimThr and windowSize depending on the sliding window reason.

And replace script "1.1_batch_run_CyDotian.py" with "3.1_batch_run_CyDotian_sliding_window.py" and "1.2_batch_run_draw_dotplot.py" with "3.3_batch_run_draw_dotplot_sliding_window.py".

For "1.3_batch_run_draw_depth_plot.py", "1.4_batch_run_output_repeat_density.py", "1.5_batch_ extract_repeat_sequences.py" and, "1.8_batch_extract_repeat_sequences_in_pairwise_comparison_mode.py ", are not available in this mode.

For batch analysis of comparison between two different sequences, use "3.2_batch_run_CyDotian_sliding_window_in_pairwise_comparison_mode.py" instead of " 1.7_batch_run_CyDotian_in_pairwise_comparison_mode.py".

Note:

If the user sees the following output statement: "Congratulations, the script worked and finished successfully!", it means that all of the user's sequences were successfully analysed.

If the user sees the following output statement: "Sadly, the script did not complete properly, please check the output log, resolve the problem, and try again!", the user has failed some or all of the sequence analysis. At this time, you can view the output message "If a successful ID appears in 'failure_sequences.txt', the number result after deleting the position file corresponding to this ID is:
This fasta file with [number] sequences has a total of [number] failures and [number] successes this time!", if the sum of the number of failed sequences and the number of successful sequences is equal to the total number of sequences entered by the user. Then the output of the user is normal. On the contrary, it is not normal.

In any case, as long as the last line of the output information is "The program running time: [number]hour(s) [number]minute(s) [number]second(s).", it means that the Python script run by the user has been executed normally.