SSR_VibraProfiler

SSR_VibraProfiler is a tool designed to establishes a DNA fingerprint database based on whole genome SSR characteristics and is used to predict variety for unknown individuals. It consists of four parts: model construction, model evaluation ,individual variety identification, and cross validation.

Requirement

- minia
- misa (only misa.pl is needed)
- python 3.9.1
- conda (The version of conda we use is 23.9.0, this does not mean that other versions of conda are definitely unusable).

Make sure they are all installed and added to the environment variables.

Installation by conda

Shell

```
conda install -c oldcat931 ssr_vibraprofiler
```

Installation by Source Code

Shell

```
git clone https://github.com/Olcat35412/SSR_VibraProfiler
cd SSR_VibraProfiler
conda env create -f SSR_VibraProfiler.yml
conda activate ssr_vibraprofiler_env
chmod +x *
#please manually add this directory to the environment variable
```

Usage

1 SSR_VibraProfiler_model_build.py

This script integrates three key steps: 1) assembly of sequencing data using Minia, 2) extraction of SSR information using MISA, and 3) screening of SSR information and model construction. Users can optionally execute the first two steps by themselves if desired.

```Shell
SSR_VibraProfiler_model_build.py -s 1 -e 3 -pp 0.8 -index index_file -o output_path
```

The following parameters are essential:

- **-s**: Specifies the start stage.
- **-e**: Specifies the end stage.
- **-pp**: Indicates the polymorphism (as described in the main text of the paper).
- **-o**: Designates the output path for storing the final identification model and SSR lists.
- **-index**: Refers to the index file.

We have two index files. The primary index file, detailed in the table below, contains two columns. The first column lists the variety for each individual, while the second column provides the corresponding absolute path for the input file of Minia. Please note that the paths are absolute paths.

Plain Text

```
 1 1    /Rho/data/W-1-01
 2 1    /Rho/data/W-1-02
 3 2    /Rho/data/W-2-01
 4 2    /Rho/data/W-2-02
 5 3    /Rho/data/W-3-01
 6 3    /Rho/data/W-3-02
 7 4    /Rho/data/W-4-01
 8 4    /Rho/data/W-4-02
 9 5    /Rho/data/W-5-01
10 5    /Rho/data/W-5-02
11 6    /Rho/data/W-6-01
12 6    /Rho/data/W-6-02
13 7    /Rho/data/W-7-01
14 7    /Rho/data/W-7-02
15 8    /Rho/data/W-8-01
16 8    /Rho/data/W-8-02
```

When Minia's input includes multiple files, an index file is used to specify the input files. This is referred to as the second-level index file. For example, the second-level index file for "/Rho/data/W-1-01" is shown in the below table. Since our files are paired-end sequencing files, each line in this file represents one end of a sequencing pair. If you are working with different types of files, you can modify the index file accordingly. If you have already completed the Minia assembly, you do not need the second-level index file.

Plain Text

```
1 /Rho/data/W-1-01_R1.fq
2 /Rho/data/W-1-01_R2.fq
```

The first-level index file is always necessary under any conditions.

The following parameters are optional, users can adjust them according to your computer configuration:

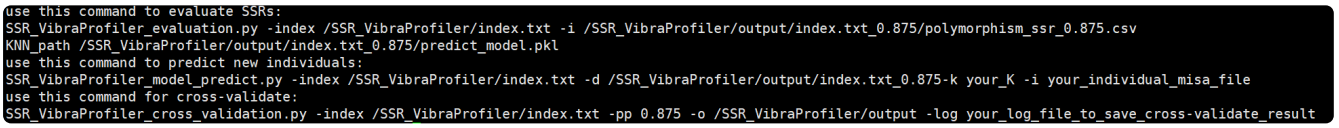
- **-misap**: Specifies the number of MISA processes to use; the default is 1
- **-miniap:** Specifies the number of minia processes to use; the default is 1
- **-miniac:** Specifies the number of cores for Minia processes; the default is 1

Next, I will explain the output files generated by this software. For example, if your index file is named "index_file" and your polymorphism indicator is "0.8" (please refer to

our main text for details), the output files include the following:

- all_data.csv;
- polymorphism_ssr0.8.csv;
- polymorphism_ssr_list.pkl;
- predict_model.pkl.

```shell
output_path/index_file_0.8/
                  └── all_data.csv
                  └── polymorphism_ssr0.8.csv
                  └── polymorphism_ssr_list.pkl
                  └── predict_model.pkl
```



> **Commands of other 3 scripts with parameters filled in**

Additionally, the output information from `SSR_VibraProfiler_model_build.py` includes the usage commands for the other three scripts, with parameters filled in.

2 SSR_VibraProfiler_evaluation.py

```shell
SSR_VibraProfiler_evaluation.py -index index_file -i polymorphism_ssr.csv
```

The following parameters are essential:

- **-index_file** Specifies the index file, which should be the same as the one used in `SSR_VibraProfiler_model_build`.
- **-i**: Specifies the input file, which is the `polymorphism_ssr.csv` generated by `SSR_VibraProfiler_model_build`.

~~~~~~~~~~~~~~~~~~~~**Update in 25.2.26**~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The following parameters are optional:
~~~~~~~~~~~~~~~~~~~~

- **-c**: Specifies the color, this file contains one RGB color per line. Please ensure that the total number of colors corresponds to the total number of varieties.

3 SSR_VibraProfiler_model_predict.py

```Shell
SSR_VibraProfiler_model_predict.py -i -d -index -k
```

The following parameters are essential:

- **-i**: Specifies the SSR information file for the individual whose variety is to be predicted.
- **-d**: Indicates the folder containing the KNN-based model file and the filtered SSR list, which are the final results from the previous step.
- **-index**: Refers to the same index file used in the first step.
- **-k**: Represents the number of nearest neighbors (k) in the KNN prediction.

Here is an example:

```Shell
SSR_VibraProfiler_model_predict.py -i W-1-10.contigs.fa.misa -d /out/index_file_6_27.txt_0.8 -index index_file_6_27.txt -k 37
```

Please note that the prediction results are not returned directly. Instead, due to variations in the number of individuals per variety within the model, the output will list the top *k* closest points to the predicted individual based on distance. For example, if *K* is set to 5, the output will display the 5 nearest points, sorted by proximity to the predicted individual.

4 SSR_VibraProfiler_cross_validation.py

```Shell
SSR_VibraProfiler_cross_validation.py -index -pp -o -log
```

The following parameters are essential:

- **-index**: Refers to the same index file used in the first step.

- **-pp**: Indicates the polymorphism (as described in the main text of the paper).
- **-o**: Output directory path.
- **-log**: File path that saved result data.

Operation example in *Rhododendron*

We used our software on 40 individuals from 8 *Rhododendron* varieties. Here are the steps to repeat our work. This can also be an example for using the software. The results information of case study can be found in https://www.scidb.cn/anonymous/dlVydXll

A more lightweight dataset is also provided. It contains only 5% of the reads from the original dataset.

1. Starting from the raw sequencing data

First, you need to download the original paired-end sequencing files. If you have downloaded the dataset that is exactly the same as the one in our paper's case study, after decompression, there will be approximately 800GB of sequencing data. If you have downloaded an alternative, more lightweight dataset, the decompressed file size will be approximately 40GB.

Here are command for download data:

Shell

```
wget "https://china.scidb.cn/download?fileId=2779cd477eecc01f550144809f6b9e9a&username=jiangchen1234@163.com&traceId=a62cbd41-3e6d-482f-a101-6c2e9d1b8967"
#Original dataset
wget https://china.scidb.cn/download?fileId=b463c9e8f1e2ce27ccc770739e662d15&username=jiangchen1234@163.com&traceId=jiangchen1234@163.com
#New lightweight dataset
```

Then you will need to follow the instructions outlined previously to prepare the minia index files for each sequencing sample, and then prepare a higher-level index file. We have already provided these index files within the supplementary materials of article; you just need to modify the corresponding absolute paths.

The commands starting from this stage are as follows:

```shell
SSR_VibraProfiler_model_build.py -s 1 -e 3 -pp 0.125 -index index.txt -o output_path
```

Note that during the minia assembly process, we utilized multi-threading, and this parameter affected the resulting contigs file, leading to inconsistencies with our assembled contigs, which in turn affects the final outcome. Therefore, we recommend downloading our contigs file directly and starting to run the program from stage 2, so that you can achieve completely consistent replicate results.

2. Starting from the contigs data

Please download our contigs data from the following link, and also prepare the index files (no longer requiring the minia index files). Here are command for download data:

```shell
wget "https://china.scidb.cn/download?fileId=fc71d3ed27011fb6ad0da74d59498989&username=jiangchen1234@163.com&traceId=a62cbd41-3e6d-482f-a101-6c2e9d1b8967"
```

Use the following commands.:

```shell
SSR_VibraProfiler_model_build.py -s 2 -e 3 -pp 0.125 -index index.txt -o output
```

After executing this script, it will generate the CSV files corresponding to the initial SSRs matrix and the filtered SSRs matrix. If u start from contigs file, the result files should be completely consistent with the ones we provided in the supplementary materials.

3. Assessment of SSRs Classification Performance

Directly use the command provided by `SSR_VibraProfiler_model_build.py`.

Your plotting results should be highly consistent with ours except for the colors. The color differences are due to the fact that the colors in the actual script are automatically generated by the program, whereas our illustrations are with specified colors.

4. LOO method evaluation

Directly use the command provided by `SSR_VibraProfiler_model_build.py`.