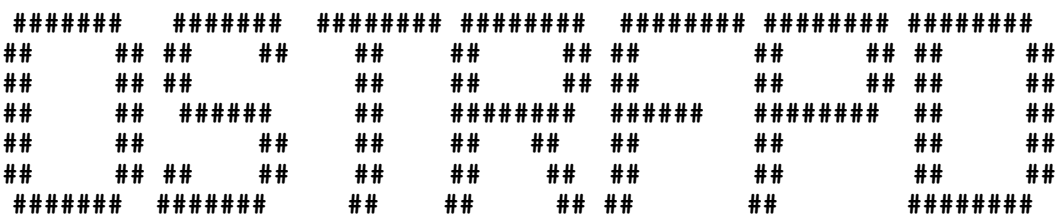

(O)mni (S)hort (T)andem (R)epeat (F)inder & (P)rimer (D)esigner v0.01

##OSTRFPD v0.01

A user friendly toolkit to identify short tandem repeats in DNA, RNA and amino acid sequences with integrated option to design microsatellite-targeted primers.

This document consists of supplementary information including installation manual, command syntax and special extended information for OSTRFPD. The main manuscript associated with the software is entitled:

"**OSTRFPD: Multifunctional tool for genome-wide short tandem repeat analysis for DNA, transcripts and amino acid sequences with integrated primer designer**"

Citation: Mathema VB, Dondorp AM, Imwong M (2019) OSTRFPD: Multifunctional tool for genome-wide short tandem repeat analysis for DNA, transcripts and amino acid sequences with integrated primer designer. Evolutionary Bioinformatics.DOI: 10.1177/1176934319843130

OVERVIEW

OSTRFPD is a versatile, platform independent open source tool written in python that enables identification and analysis of genome-wide short tandem repeats (STRs) in nucleic acids (DNA and RNA) and protein sequences supplied as FASTA formatted files. OSTRFPD is designed to run with full-features in both command line interface (CLI) or graphical user interface (GUI) mode based on users' requirement. OSTRFPD can detect both perfect and imperfect repeats of low complexity with customizable scores. The software has built-in architecture to simultaneously refine selection of flanking regions in DNA and automatically generate microsatellite-targeted primers implementing Primer3 platform as a single step integrated operation. The software has built-in fast and accurate motif generator engines and additional option to use dictionary mode (optimized for searching longer repeats). Dictionary file containing common deoxynucleotide and amino acid motifs observed in *Plasmodium* species has been bundled together with the software. Completion of each search generates result along with general statistics containing motif categorization, repeat frequencies, densities, coverage, %GC (both source and microsatellites) content and simple text-based imperfect alignment visualization. The implementation of OSTRFPD is demonstrated using publicly available whole genome sequence of selected *Plasmodium* species.
The Software and source code is freely available at: https://github.com/vivekmathema/OSTRFPD

PREFACE

Although thorough attention has been given during design and development of OSTRFPD, current version of the software may still contain some minor bugs and compatibility issues. This effect is common to early stages of several other software release and distribution. However, open source nature of this software enables it to be corrected, adopted and further optimized by the global community. We will be highly encouraged for future upgrades and further contribution based on positive feedbacks from our users.

DESTRIBUTION, DEPENDENCIES AND INSTALLATIONS

The details of the software aims and architecture is described in the associated paper. We here focus on installation and command syntax.

OSPTFPD is an open source software distributed under **General Public License (GPL v 3)**. Apart from standard python modules, the software imports some functions from the dependencies under the 'Terms and Conditions' allowed under by the standard license.

In addition to Python (version 3.5 or above) the dependencies mentioned in 'requirements.txt' can be installed using standard 'pip' or 'pip3' installer for python.

Both source and binaries of OSTRFPD have been successfully tested in Windows 7, 10 and Linux Ubuntu 16.04 with correctly installed dependencies.

==========

IMPORTANT NOTICE:

- WE RECOMMEND USING WINDOWS VERSION OF THE PYTHON SOURCE CODE AND PRE-COMPILED BINARIES FOR TESTING. SINCE LINUX SYSTEMS ARE HIGHLY CUSTOMIZED AND PRE-BUILT BINARIES OF OSTRFPD AND PRIMER_CORE MAY NOT WORK PROPERLY LEADING TO MALFUNCTIONING OF THE SOFTWARE.
- FOR MAINTANING CONFIDENTIALITY, THE MAIN SOURCE CODE AND BINARIES ARE PROTECTED BY PASSWORD UNTILL MANUSCRIPT IS ACCEPTED FOR PUBLICATION. THE PASSWORD ARE CASE-SENSITIVE AND ARE GIVEN AS "PASSWORD" in "Availability of Data and Materials" SECTION OF THE SUBMITTED MANUSCRIPT FOR REVIEW]

- FOR PROPER VIEW OF TEXT-BASED ALIGNMENT VIEW, WE REQUEST TO USE THE FONT "CONSOLAS" IN TEXT VIEWER LIKE "NOTEPAD". IF NOT AVAILABLE, THE RECOMMENDED FONT CAN BE FREELY DOWNLOADED FROM THE LINK: https://www.wfonts.com/font/consolas (NORMALLY INSTALLED BY DEFAULT IN MOST OPERATING SYSTEMS AND CAN BE CHOSEN UNDER FONTS OPTION).

- TO OVERCOME SIZE LIMITATION AND EASE OF DOWNLOAD, SOURCES AND BINARIES OF OSTRFPD IS SHARED IN GOOGLE DRIVE LINKS GIVEN BELOW:

==========

DOWNLOAD LINKS FOR FULL SOURCE CODE AND PRE-COMPILED BINARIES

1) WINDOWS PACKAGE (full source code + pre-compiled binaries + example fasta files):

Link: https://drive.google.com/file/d/1GhInYTiBIOLkRmUBL3HZSSGLce0L_NQV/view?usp=sharing

2) LINUX PACKAGE (full source code + pre-compiled binaries + example fasta files):

Link: https://drive.google.com/file/d/1qpTsiR07Uru5S_YcKAI6voPxbYKeC-5W/view?usp=sharing

INSTALLATION

```
Install Python3 from https://www.python.org/downloads/release/python-350
```

Whereby you should be able to download and install 32 or 64 bit version of Python 3.5. The 64 bit version is recommended

Depending upon your python version use of ‘pip’ or ‘pip3’ should be fine to install necessary dependencies. Please follow 'upgrade pip' instruction if you need to upgrade your pip for latest version and compatibility issues.

```
**pip3 install PyQt5==5.9.1**
```

This should result in installation of PyQt5 plugin required for GUI of OSTRFPD.

```
**pip3 install biopython==1.72** or pip3 install biopython version 1.72
```

This should result in installation of biopython and its sub-dependencies

Likewise, installations of futures for python version syntax compatibility

```
**pip3 install future==0.16.0**
```

(OPTIONAL AND EXPERIMENTAL) For windows

```
**pip3 install pyinstaller==3.3.1**
```

This will install pyinstaller which can be used to generate standalone OSTRFPD binary from source for windows using following command in console.

```
**pyinstaller ostrfpd.py --ONEFILE**
```

This should generate standalone 'ostrfpd.exe' binary in ‘\OSTRDPF\dist’ folder for windows. Warning! The compilation can only be possible if all system environmental variables and sub-dependencies are supported. May not work is some installations due to library version incompatibility (mostly PyQt5-related) of sub-dependencies during compilation for creating standalone binaries. In order to resolve the issue, a pre-compiled windows binary of OSTRFPD has been supplied along with the source code release.

EXECUTION

From here onwards, we assume that users have downloaded and extracted all content into 'OSTRFPD' folder which should at least contain 'ostrfpd.py', 'primer3_core' (optional plugin for primer creation), binaries (optional'ostrfpd.exe' for Windows or Linux 'ostrfpd' binary for Linux) and dictionary files (optional) for the software to work properly.

```
##TYPICAL STRUCTURE OF THE OSTRFPD FOLDER

OSTRFPD_v0.01
|
|-----ostrfpd.py
|-----gui_interface.py
|-----ostrfpd.exe              [ Note: 'ostrfpd' for LINUX         ]
|-----primer3_core.exe         [ Note: 'primer3_core' for LINUX ]
|-----run_ostrfpd_bin_gui.bat
|-----run_ostrfpd_src_gui.bat
|-----dict_dna.txt
|-----dict_protein.txt
|-----input.fasta
|-----requirments.txt
|-----run_info_readme.txt
|-----LICENCE
|-----rna_seq                  (Note: subfolder)
      |-----ribosome1_seq_URS0000857690.fasta
      |-----ribosome2_seq_URS0000A6B25D.fasta
```

OSTRFPD can be run in either user-friendly GUI mode or command line interface (CLI).
Please Note that all parameters or arguments are case sensitive.
The GUI mode can be initialized by simple argument "-gui true".[i.e. python3 ostrfpd.py –gui true]

IN WINDOWS:

Open windows console (can be initiated by typing cmd.exe in start box) and type the commands (case sensitive).

```
[from source code]
python3 ostrfpd.py -gui true

[from binaries]
ostrfpd.exe –gui true
```

This should launch OSTRFPD in user-friendly GUI mode containing built-in helper tooltip text, self-explanatory buttons and basic level of error handling interface.

Same steps can be achieved by just running the 'run_ostrfpd_binary.bat' or 'run_ostrfpd_source.bat', which

will run the supplied binary (.exe) or source (.py) directly in GUI mode without the requirement to enter console mode.

The OSTRFPD is supplied with untampered primer3_binaries (version 1.1.4) for windows (primer3_core.exe) which can also be independently downloaded and/or compiled from the official source (https://sourceforge.net/projects/primer3/files/primer3/1.1.4/) following the creators' instructions.

IMPORTANT: Please note that both OSTRFPD binaries (and source) and primer3_core.exe should be in same location even when using pre-compiled ostrfpd.exe (Refer to: Typical Structure of the OSTRFPD Folder).

IN LINUX

Open Linux Terminal and type the command (Warning! All parameters or arguments are case sensitive).

```
$ python3 ostrfpd.py -gui true
```

This should launch OSTRFPD in user-friendly GUI mode containing built-in helper tooltip text, self-explanatory buttons and basic level of error handling interface.

The OSTRFPD is supplied with non-tampered pre-compiled Primer3 (ver 1.1.4) binaries named as 'primer3_core' for Linux which can also be independently downloaded and compiled from the source (https://sourceforge.net/projects/primer3/files/primer3/1.1.4/) following the creators' instruction.

We recommend the Linux users to compile primer3_core from source. However, OSTRFPD is also bundled with pre-compiled 'primer3_core' binary, just in case for ease of use.

IMPORTANT: Please note that both OSTRFPD binaries (and source file 'ostrfpd.py') and 'primer3_core' binary should be in same location (Refer to: Typical Structure of the OSTRFPD Folder).

Typically for Linux users (here, we used Ubuntu 16.04) the system PATH variable should be set to the location from where 'primer3_core', 'ostrfpd.py' or equivalent binary is expected to be run. A typical example is given below. In terminal with likely use of 'sudo previliges' type your path where 'ostrfpd.py' files are located.

```
export PATH=/home/user/Documents/OSTRFPD/:$PATH
```

Optionally, if users chooses to compile the ostrfpd.py from source using LINUX version of pyinstaller (Refer to pyinstaller installation for Windows) and then use the binary, please follow the command below from OSTRFPD directory to generate LINUX binaries.

```
$ pyinstaller -D -F -n ostrfpd --onefile -c "ostrfpd.py"
```

This should create linux compatible binary 'ostrfpd' in /OSTRFPD/dist folder, which can then be copied to the base OSTRFPD folder containing 'primer3_core' binary for use.

##SETTING UP THE BINARIES FOR EXECUTION IN VIA LINUX TERMINAL.

Necessary ***sudo*** user privileges may have to be acquired depending upon system administrative settings.

After compilation or to directly use the supplied pre-built binaries (both ostrfpd and primer3_core) the users may have make the file executable using:

```
**chmod +x app-name** (e.g.: 'chmod +x ostrfpd' and 'chmod +x primer3_core') command and then execute the ostrfpd binary from terminal as:

**terminal$ *./ostrfpd***

```

Once the binaries are ready for execution, similar to Windows, launch the OSTRFPD (for graphical interface) using:

(i) From source

```
**$ python3 ostrfpd.py -gui true**
```

(ii) From supplied binary

```
**$ ./ostrfpd -gui true**
```

Note: Please refer to the Typical Structure of the OSTRFPD Folder for maintaining correct folder structure of OSTRFPD.

BARCHART DISPLAY FOR THE CATAGORIZED MOTIF FREQUENCIES

This EITIRELY OPTIONAL segment of the code allows interested advance users to directly generate bar charts of the motif frequencies for visualization using additional 'matplot' and 'numpy' library of the python.

Most distribution of Matplotlib and numpy should be compatible with this code. Typically follow installation for dependencies should work file:

```
**pip3 install matplotlib==2.2.2** [tested for matplot version 2.2.2 ]

**pip3 install numpy==1.14.5**  [ tested for numpy version 1.14.5]

```

To activate the bar chart drawing module in code, please uncomment following sections in the ostrfpd.py source code.

i. The function block for "def motif_chart(lst):"
ii. "motif_chart(retval) if retval != [] else None" in [__name__ == "__main__"]
iii. "motif_chart(retval) if retval != [] else None" in [def GUI_module(self):]

Note: We do NOT recommend to compile the OSTRFPD to obtain standalone binaries if this Chart drawing function is used.

OSTRFPD GRAPHICAL USER INTERFACE (GUI) MODE

The GUI mode can be accessed for both windows and Linux systems as described earlier.

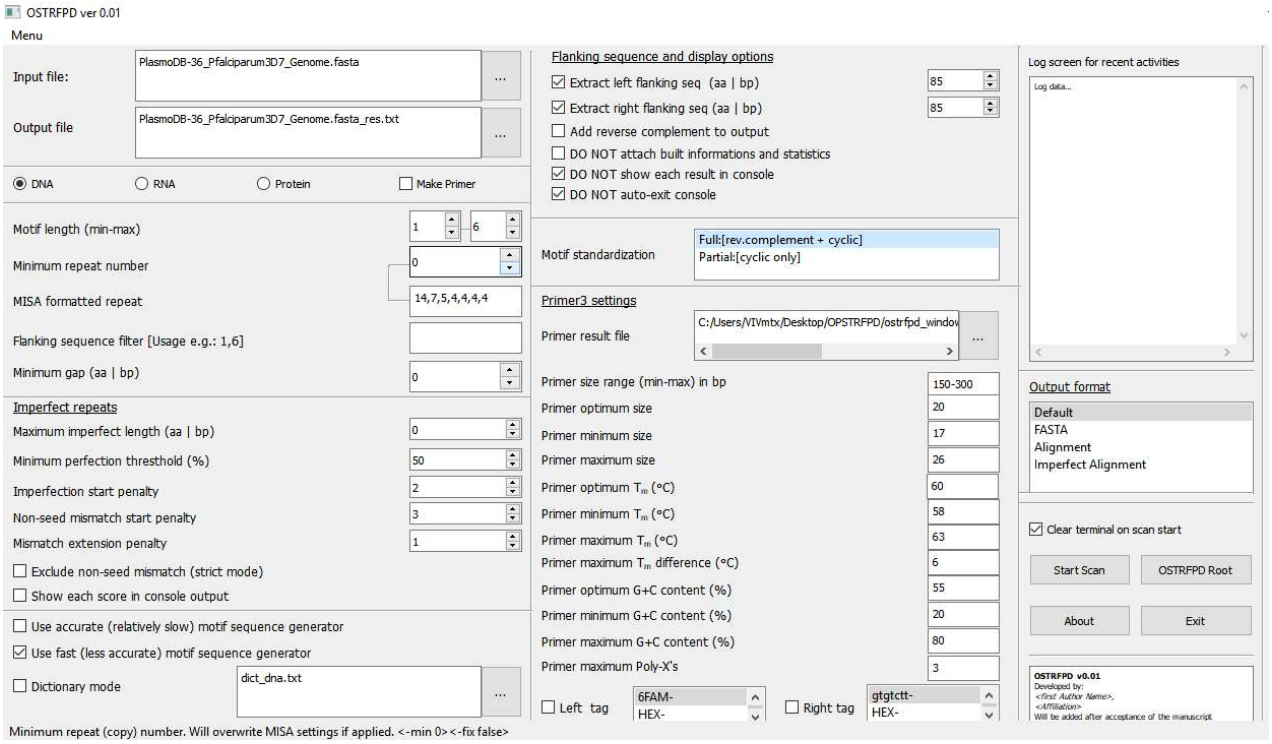

Figure 1. Screenshot of OSTRFPD running in GUI mode for nucleotide repeat. The screenshot shows default options used for searching 1-6 bp microsatellites with minimum repeat of 14, 7,5,4,4 and 4 for unit motif length of 1, 2,3,4,5, and 6, respectively. Please note that the helper tool tip text with brief information regarding value currently under pointer is provided on bottom status bar of the window. Equivalent command line code is 'python3 ostrfpd.py –scan dna -input source_fastafile –unitmin 1 –unitmax 6 –misa 14,7,5,4,4,4'

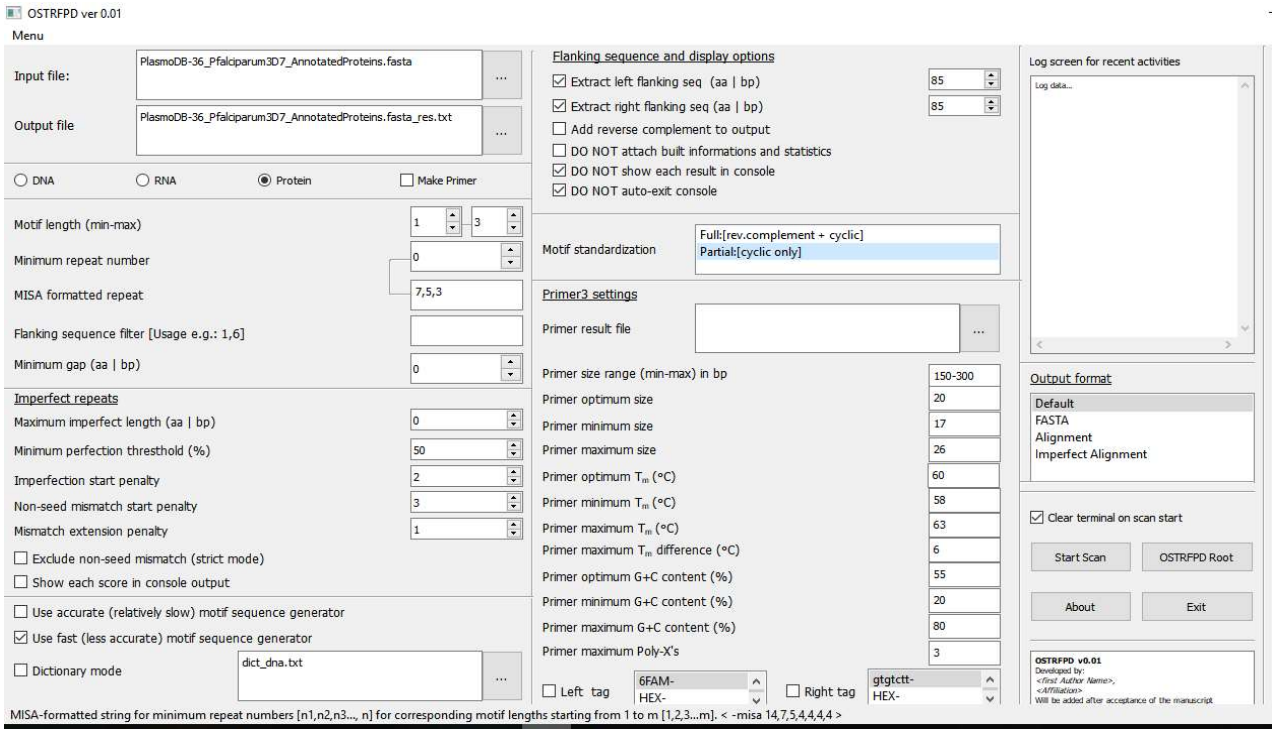

Figure 2. Screenshot of OSTRFPD running in GUI mode for searching amino acid repeat. The screenshot shows default options used for searching 1-3 aa motif with minimum repeat number of 7,5 and 3 for unit motif lengths of 1, 2 and 3, respectively. Please note that the helper tool tip text with brief information regarding value currently under pointer is provided on bottom status bar of the window. Equivalent command line code is 'python3 ostrfpd.py –scan protein -input source_fastafile –unitmin 1 –unitmax 3 –misa 7,5,3'

OSTRFPD COMMAND LINE INTERFACE (CLI) MODE

The CLI provides a full-fledge control over OSTRFPD operation. CLI interface allows OSTRFPD to be used as plugin for batch operation by other software. However, users should be careful during arguments or parameter input as these are case sensitive.

Arguments can be in any order but should avoid supplying conflicting or repeated arguments. Unlike the GUI mode, there is no auto-range or auto-correction and other error handling modules in CLI mode. Thus, users must be strictly follow guidelines and use parameters accordingly.

Since GUI mode has almost full-fledge features in comparison to CLI mode, whenever possible we highly recommend to use GUI mode.

SCREENSHOT OF OSTRFPD RUNNING IN CLI MODE

Figure 3. Screenshot of OSTRFPD running in CLI mode for identification of microsatellites. The screenshot shows default options used for searching 1-6 bp microsatellites with minimum repeat of 14, 7,5,4,4 and 4 for unit motif length of 1, 2,3,4,5, and 6, respectively. Please note that the command arguments to generate output can be seen on the console window title bar. The general statistics shown on bottom of the screen is also written to the result output file by default.

CLI COMMAND SYNTAX AND ARGUMENTS:

A list of arguments and their description is given below:-

The "--help" arguments will give a standard python help for all command syntax and associated description. Except for the python default argument “--help’, all of the OSTRFPD CLI arguments beings with single negative (-) sign followed by argument name (i.e.: -argument) and the numeric values are represented by uppercase word of the argument.

Whenever feasible, a short description and example of default CLI argument is shown in helper tool-tip message on status bar in GUI mode as: “< -argument value >”.

```
Usage: ostrfpd.py [-h] [-input INPUT] [-output OUTPUT] [-unitmax UNITMAX]
                  [-unitmin UNITMIN] [-min MIN] [-imperfect IMPERFECT]
                  [-lflank LFLANK] [-rflank RFLANK]
                  [-exclude {None,true,false}] [-imop IMOP] [-mop MOP]
                  [-scan {None,dna,rna,protein}] [-mmp MMP]
                  [-rcomp {None,true,false}] [-misa MISA] [-eng {true,false}]
                  [-sshow {true,false}] [-fsc FSC] [-pfname PFNAME]
                  [-fix {true,false}] [-primer {true,false}]
                  [-std {None,true,false}] [-dict DICT] [-fasta {true,false}]
                  [-sdout {true,false}] [-gap GAP] [-sim SIM]
                  [-stats {true,false}] [-report {true,false}]
                  [-align {None,true,false}] [-autoexit {true,false}]
                  [-ltag LTAG] [-rtag RTAG] [-imalign {None,true,false}]
                  [-gui {None,true,false}] [-prng PRNG] [-posz POSZ]
                  [-pmisz PMISZ] [-pmxsz PMXSZ] [-potm POTM] [-pmitm PMITM]
                  [-pmxtm PMXTM] [-ptmdiff PTMDIFF] [-pogc POGC]
                  [-pmigc PMIGC] [-pmxgc PMXGC] [-pmpoly PMPOLY] [-v V]
```

Description of each arguments:

-h, --help Shows help message and exits the program.

-input INPUT Full pathname of input file (i.e. source FASTA file). If not supplied, the "input.fasta" file will be searched by default. Single, multi-FASTA and gunzip-compressed (.gz) FASTA are supported for direct scan without unzipping.

-output OUTPUT Full pathname of output result file. If not supplied, the <source filename + "_res.txt"> will be used as default output report file. The default output file is tab-delimited plain text file containing multiple parameters (e.g.: unit motif, repeat number, start stop position, flanking sequences). The output file can also be configured to contain built information and general motif statistics and categorization.
Note: the file can be imported in Microsoft excel by changing the file extension to .xls for further processing.

-primer {true,false} Sets the flag to make primer using Primer3. Default value is false.

-pfname PFNAME Full pathname of primer result output file to be saved. If not supplied, by default the output name will be used as [source filename + ‘_prm.txt’].The output is a tad-delimited plain text file. The primer output file will contain Sequence IDs of primer derived from the original sequences from which the primers were designed. In addition to left (forward)

and right (reverse) primers, the corresponding standardized unit motifs, motif copy number, T_m and complete sequence (Left flanking+ microsatellite + right flanking) will be tabulated in the primer result file.
Note: the file can be imported in Microsoft excel by changing the file extension to .xls for further processing.

-scan {None, dna, rna, protein}
Sets the expected source sequence type to DNA, RNA, or PROTEIN. Default value is set to dna.

-unitmax UNITMAX Input type: positive integer. Sets the maximum unit motif length to be searched.

If used with '-min' argument only MIN single fixed unit will be scanned.

In Fast (default) search mode, the maximum values for DNA,RNA and Amino acid unit motifs are 10, 10 and 3 bp|aa respectively.

In Accurate search mode, the maximum values for DNA, RNA and Amino acid unit motifs are 6, 6 and 3 bp|aa respectively.

In Dictionary-based search mode, the maximum values for DNA, RNA and Amino acid unit motifs is 30 bp|aa.

-unitmin UNITMIN Input type: positive integer. This option sets the minimum unit motif length to be searched. Default values is 1. The UNITMIN can range from 1 to UNITMAX.

If UNITMIN and UNITMAX are same, then single fixed length unit motif will be search. Use of '-fix true' argument will overwrite the '-unitmin UNITMIN' setting and forcefully set the UNITMIN = UNITMAX (i.e. single fixed unit motif only).

-min MIN Input type: Positive integer. Sets the minimum repeats (copy number) threshold for selection of tandemly repeated sequences. This option will overwrite MISA-formatted minimum repeat settings (if present). By default the MIN value is 0 (i.e. OFF).

-fix {true,false} Sets a single fixed minimum repeats (copy number) for all unit motifs. This option will overwrite MISA settings (if present). Default value is false.

-misa MISA MISA-formatted number series to input different minimum repeat number for different motif length. [e.g: -misa 14,7,5,4,4 for minimum repeat number of microsatellites with unit motif length of 1,2,3,4 and 5, respectively]. Note that the minimum repeat values for misa should be supplied as '1,2,3,...,UNITMAX-1,UNITMAX' even if minimum unit motif length (UNITMIN)>1. The misa is automatically set as NULL if '-min' > 0. Default value of misa string for DNA or RNA is '14,7,5,4,4,4' and Protein is '7,5,3,3', respectively. Whenever possible do avoid using '-misa' and '-min' arguments together to make syntax clear for operation.

-lflank LFLANK Input type: Positive integer. Extracts the left flanking sequence. This option auto-truncates the flanking sequence region if the left side sequence is smaller than supplied length (bp | aa) of flanking sequence. Default value is 85.

-rflank RFLANK Input type: Positive integer. Extracts the right flanking sequence. This option Auto-truncates the flanking sequence region if the right side sequence is smaller than supplied length (bp | aa) of flanking sequence. Default value is 85.

-fsc FSC Used as positive integers in the format 'm,n' (Typical use: '-fsc 1,5' or '-fsc 1,6'). Default

status is OFF or None. Filters out the microsatellites if a minimum of 'm' unit motif length tandem repeat of minimum 'n' repeat number (copy number) is found in Left or Right flanking sequence. Note: The option is useful for removing microsatellites from results whose low-numbered repeats may cause problem in primer design.

-eng {true,false} Sets the unit motif sequence generator engine type for accurate fast or accurate search. Default value is 'false' for fast but less efficient motif pattern search. If set 'true' then uses slow yet accurate engine.

Fast engine is recommend to use if required unit motif length ≤10 for nucleic acids and ≤3 for proteins.

Accurate engine is recommend to use if required unit motif length ≤6 for nucleic acids and ≤3 for proteins.

-std {None,true,false} Standardizes the unit motif (seed) name for display and report in either Partial:[cyclic equivalent motifs] or Full: [reverse complement + cyclic equivalents]. Default is set to Full standardization. See citation paper for details. General statistics results for motif characterization will also be formatted based on the type of motif standardization. For Protein sequence, as there are no complement strands, only partial standardization is utilized for amino acid unit motif characterization.

-imperfect IMPERFECT

Maximum distance (aa|bp) within which the mismatch (indel) containing tandem repeats but with same unit motif (seed) will be combined as single imperfect repeat. The value for argument '–imperfect' > 0, automatically turns ON search for imperfect repeats as associated default score values [i.e. –imop, -mop, -mmp]

-exclude {None, true, false}

Restricted or Strict mode. Ignores non-seed (aa|bp) containing imperfect microsatellites even if the non-seed mismatch occurred within pre-declared imperfection range. Default value is None or false.

-imop IMOP Input type: positive integer. Penalty score for the first non-seed mismatch occurrence (added only once for an imperfect microsatellite independently on top of '-mop' and '-mss' parameter values). Default value is 3 and only applied if value for '-imperfect' > 0 (i.e. IMPERFECT mode is set ON).

-mop MOP Input type: positive integer. Penalty score which is applied only once for the first (starting mismatch) occurrence to a given imperfect microsatellite. Default value is 2 only applied if value for '-imperfect' > 0 (i.e. IMPERFECT mode is set ON).

-mmp MMP Input type: positive integer. Set the penalty for each mismatch (seed or non-seed type). It is added on top of mismatch starting penalties supplied by '-impo' and/or '-mop' if present. Default value is 2 and only applied if value for '-imperfect' > 0 (i.e. IMPERFECT mode is set ON).

-rcomp {None,true,false}

Attaches the reverse complement sequence for DNA or RNA repeats including flanking sequence. Default value is None or false.

-sdout {true,false} Sets the standard result display flag for console or terminal. If true, displays each identified repeat result on console or terminal screen. If set false, only the Sequence IDs being search along with cumulative frequency (C.Freq | total |) of identified repeats and primer designed (→ Total primers designed) will be shown. Has no effect on output

result file. If False, Default value is false.

-sshow {true,false} Sets the display output ON/OFF for the scoring matrix on console screen during scan for each tandem repeats identified. Has no effect on output result file. Default value is false. This option is forced turn OFF if '-sdout' is false.

-dict DICT [Usage: -dict filename]. Dictionary file is a plain text file contaning list of unit motifs of same type molecules (DNA, RNA, or amino acids) each separated by new line. Dictionary file is used to exclusively search custom user-supplied unit motif sequences. If used, only the unit motifs listed in the dictionary file (e.g. dict_dna.txt) will be searched for a single fixed (default: -fix true) length (default: -unit 8) unit motifs of maximum upto 30 bp and fixed minimum repeat number (default: -min 4). This mode [-dict] is expected to be used together with [-unit, -fix and -min] parameters.

Extended Information for dictionary mode special usage: During runtime, all duplicates or equivalent (cyclic as well as complementary) motifs from supplied dictionary file will be filtered out and only unique standardized motifs will be used for searching. This mode may also be used to roughly estimate density of certain oligonucleotides (14-30 bp) or signaling peptides (14-30 aa). In case of oligonucleotide (e.g.: transcription factor binding sequence) or special short peptide sequences (e.g.: signaling peptide), a fixed minimum repeat value of 1 (i.e. '–min 1') may be used to roughly quantify their genome- or proteome-wide occurrence or densities. However, the feature is experimental and future improvements will be done accordingly. We recommend using GUI mode for easy operation of this feature.

-align {None,true,false}
Output the report file in alignment form (not applicable for FASTA or Custom format).

-fasta {true,false} Sets the output result to FASTA formatted file. File description, motif statistics or built headers is NOT available for this Mode. If output filename is not supplied source filename + '_res.txt' will we written as default output. Default values if false.

-imalign {None,true,false}
Make the alignment output file only contain imperfect alignment results after satisfying all other applicable conditions. The alignment option will be automatically activated. Default value is None or false.

-gap GAP Input type: Positive integer. Sets minimum gaps (aa | bp) between two consecutive repeat motifs. Please note that all consecutive repeats identified within the range will be discarded even if it belongs to same motif type. Default value is zero. The default value is 0.

-sim SIM Input type: positive integer. Sets the minimum similarity threshold value (in percentage, e.g: -sim 50 for 50%) for the results to be accepted on top of other selection criteria supplied. Similarity basically refers to percentage of complete unit motifs present in the imperfect repeat sequence compared to its near equivalent perfect repeat. All perfect repeats have similarity value of 100% (See citation paper for details). Default value is 50.

-stats {true,false} Displays and append basic statistics of the results to output file (option not applicable for FASTA output)

-report {true,false} Appends Built header on top and Basic statistics report at the end of the output file (not

applicable for FASTA output)

-autoexit {true,false}
Sets auto exit flag. If true, auto exits after completion of task or pauses the console screen if false (applicable mainly for windows system GUI launcher). Default value is false.

-ltag LTAG Attaches the left tag (e.g: 6FAM-) for the left primer generated by Primer3. Typical use ('-ltag 6FAM-'). Default value is None.

-rtag RTAG Attach the right tag (e.g: gtgtctt-) for the right primer generated by Primer3. Typical use ('-rtag gtgtctt-'). Default value is None.

-gui {None,true,false}
Open the OSTFRPD in GUI interface to input the configuration parameters. Using this option will OVERWRITE all other CLI arguments supplied and launches OSTRFPD is fresh default GUI mode. Default value is false.

-prng PRNG Input type: String. Used to set min-max (e.g: 150-300), the minimum and maximum range (in bp) of the output primers product (amplicon size). Default value is 150-300.

-posz POSZ Input type: positive integer. Used to set the optimum primer length (in Bp). Default value is 20.

-pmisz PMISZ Input type: positive integer. Used to set the minimum primer length (in Bp). Default value is 17.

-pmxsz PMXSZ Input type: positive integer. Used to set the maximum primer length (in Bp). Default value is 26.

-potm POTM Input type: positive integer. Used to set the optimum primer Tm (°C). Default value is 60.

-pmitm PMITM Input type: positive integer. Used to set the minimum primer Tm (°C). Default value is 58.

-pmxtm PMXTM Input type: positive integer. Used to set the maximum primer Tm (°C). Default value is 63.

-ptmdiff PTMDIFF Input type: positive integer. Used to set the maximum primer Tm (°C) difference. Default value is 6.

-pogc POGC Input type: positive integer. Used to set the optimum GC content (given in percentage). Default value is 55.

-pmigc PMIGC Input type: positive integer. Used to set the minimum GC content (given in percentage). Default value is 20.

-pmxgc PMXGC Input type: positive integer. Used to set the maximum GC content. Default value is 80.

-pmpoly PMPOLY Input type: positive integer. Used to set the maximum Poly-X's in primer. Default value is 3.

-v V Shows the current version information and basic information regarding the authors, information for citations, and software repository for OSTRFPD. (This section will be finalized upon manuscript acceptance)

==========

Dr. Vivek Bhakta Mathema

Source code maintainer
Please feel free to get in touch with comments, suggestions and any form of queries. Your feedback will help us improve OSTRFPD and assist researchers to achieve their goals.
Best wishes,