SSR Finder manual

Author: Jonathan A. Shortt and David D. Pollock

Version: 3.52

Manual updated on 13 December 2019

Contact: jonathan.shortt@cuanschutz.edu

License:

Background:

SSR Finder is a program written in perl that is used to locate simple sequence repeats (SSRs) and SSR-derived regions in a fasta sequence, using clouds (or groups) or related and over-represented kmers. The program operates as part of a suite of perl modules, other scripts, and a control file system which are all required for it to run. These dependencies are:

control, default, factory, SSRTools.pm, seqTools.pm, basics.pm, globals.pm, kClouds.pm, cloudPos[1], and MultiThreadCommands.pl, SSRCloudBuilder.pl

Bedtools[2] must also be installed on the computer.

The software accompanying this manual is a beta release- it has not been extensively tested in different environments. This manual describes basic parameters and operations to use the SSR Finder but does not exhaustively cover each topic at this time. If you would like to use the software and have questions about operation, please contact me at jonathan.shortt@cuanschutz.edu to request support; I'd be delighted for others to use the program and am happy to work through some problems. Additionally, these requests will allow me to gain insight into common problems with the software and manual and will thus enable better support in the manual and in the software itself.

Citations:

If you use this program, please cite our preprint:

Finding and extending ancient simple sequence repeat-derived regions in the human genome
Jonathan A. Shortt, Robert P. Ruggiero, Corey Cox, Aaron C. Wacholder, David D. Pollock
bioRxiv 697813; doi: https://doi.org/10.1101/697813

[1] cloudPos is provided as a script written is C. This needs to be compiled before it can be used as part of SSR Finder. It may also need to be made executable which can be accomplished using the chmod command (chmod u+x cloudPos).

[2] Instructions for download can be found at: http://bedtools.readthedocs.io/en/latest/content/installation.html

General Usage:

In order to run SSR Finder, make sure that all files mentioned above are in the directory where SSR Finder will be called. SSR_finder_vXXX.pl must also be in the directory, and perl must be installed on the computer. To run SSR Finder, navigate to the directory where the program is saved in a terminal window and type

perl SSR Finder_v3.51.pl --runmode [annotatePerfects | combinePerfects | buildClouds | combineClouds | annotateClouds | exportFPR] --dir [directory where files should be saved]

followed by the other arguments (args) used to specify how SSR Finder will operate within the specified run mode. Each arg supplied at the command line follows this (pretty standard) format: "--ARGNAME ARGVALUE", without quotation marks, and multiple args are separated by a space between the preceding arg's value and the next arg's name. See the Arguments section within each run mode of this manual for more information on which args must be supplied to each runmode.

Default args are contained within the default and control files. Some of these args must also be supplied in the command line. All args in the default file are also included in the control file or given on the command line. The default file provides default parameters to be used if a parameter is not supplied by the user in the control file or on the command line. Thus, parameters in the control file supersede those in the default file, and parameters that are passed in as args at the command line supersede both. I highly recommend that the default file never be edited. Instead, a number of parameters can be passed at the command line, and others are easily modifiable in the control file.

Parameters and Arguments:

Below is a list of parameters that are specified within the control file. Format for each parameter is: Name: value type [acceptable range, if applicable], default value (set in "default" file), run modes that use the parameter, and a description of the parameter.

motifmax: integer [1-6], 6, all run modes, maximum length of SSR motifs to be created/analyzed. All motifs from length 1 to motifmax are created.

rev_com: binary, 0|1, all run modes, whether to include reverse complements as part of a motif family (1), or not (0)

oligolength: integer [2-16], 12, all run modes, specifies the length of kmers that will be used when searching for perfect SSR kmers.

motifs_of_interest: all run modes, specifies which SSRs to use in a given run mode. Leave blank to use all SSR motifs. To use multiple motif families, separate each family by a space.

thread_count: integer [1-50], 4, used in annotatePerfects, specifies the number of processors that might be used by a computer at the same time. This number should never exceed the number of processors on the computer.

core_set: integer, 4, used in buildClouds, sets the most stringent core threshold as (number of loci used for clouds building)/(core_set). For example, under the default setting of 4, a family with 1,000 loci would have its most stringent threshold set to 250. A family with 4,000 loci would have its most

stringent cloud threshold set to 1,000. Likewise, if the core-set value was set to 5, these families would have threshold counts of 200 and 800, respectively.

shell_factor: float, NOT USED

cloud_info_file: string, cloud_info_file, used in buildClouds, name of file where summary information about the clouds for each SSR motif will be printed

slop_len: integer, 50, used in buildClouds, specifies how many base pairs on either side of a locus will be used as template for cloud formation

cloud_kmer_length: integer, 16, used in buildClouds, specifies the length of kmers to be used in cloud formation

cloud_strin_cutoff: integer, 5, used in combineClouds, specifies the lowest stringency kmers to use in including kmers. Kmers belonging only to clouds with lower stringency (with 1 being most stringent) than this number will be excluded.

max_strin: integer, 5, multiple run modes, the maximum number of stringency levels, also the least stringent level

max_fams: integer, 50, annotateClouds, the maximum number of families that will be printed at each locus

seed: integer, 248598, , sets a random number seed, not really used

loudness: integer [1:5], 2, sets the volume of program output to terminal screen

record_loudness: integer [1:5], 2, sets the volume of program output printed to record file

errorout: string, ErrorLog_SSR_finder.txt, sets the file name for file where errors are printed

record_run: string, RunRecord_SSR_finder.txt, sets the file name to record information about each program run

merge: integer: 0, , distance between neighboring loci at which loci will be merged into a single locus

Run modes:

SSR Finder has several different run modes, each used to run a separate part of an SSR annotation pipeline. Each of these run modes, their necessary parameters, and a brief explanation of the run mode are described below.

annotatePerfects

Necessary command line args: runmode, dir, fasta

Other args (modified in control file): File_count, motifs_of_interest, rev_com, motifmax, oligolength

Description: This run mode is used to find all perfect SSR loci (SSRs without any mutation within the locus). The run mode will create two different directories in the output directory: PERFECTS_KMERS, and PERFECTS_BEDS. The former contains a file for each motif of interest listing the kmers used to search for

repeats of that family in the provided fasta. The latter directory contains bed file-like files specifying the genomic coordinates of each family found.

combinePerfects

Necessary command line args: runmode, dir, out

Other args (modified in the control file): merge, motifs_of_interest

Description: This run mode is used to combine the annotations made in annotatePerfects into a single file. Loci can be merged within a user-specified separation, though this will remove information on the identity of the motif families that reside throughout the file. Without setting a merge distance, loci from the motifs of interest are combined into a single file, and their identities will be preserved however some overlap between loci may occur.

buildClouds

Necessary command line args: runmode, dir, fasta, genome

Other args (modified in control file):

Number of cloud layers per family

Cloud core thresholds

Motifs of interest

Description: Builds kmer clouds using specified parameters. Makes clouds for every family of interest using the bed files created in annotatePerfects.

combineClouds

Necessary command line args: runmode, dir, out

Other args (modified in control file):

Motifs of interest

Number of cores

Description: Combines kmer clouds from multiple motifs into one file. Kmers are printed for every time they appear in a file, so kmers may appear multiple times in the combined file.

annotateClouds

Necessary command line args: runmode, dir, mode, fasta, FPTable, assign, out

Other args (modified in control file):

Merge

Description: finds the location of all kmers in the user specified assign file and prints to file. Two modes are applicable within this run mode: sim and hg. In the sim mode, loci are treated as if found in a simulated genome and no false positive rate is shown for each. As such, the FPTable argument must be present but is not used in this mode. The second mode, hg38, finds locations of all kmers from the specified assign file and uses information from the provided FPTable to give a false positive rate to each

locus. As such, a file with false positive rates for different possible locus lengths must be provided. See run mode exportFPR for more on this.

The first three columns of the output of annotateClouds follows standard .bed file format (chromosome, begin, end), however some of the remaining fields require special explanation.

The fourth field contains a locus false positive rate. This field will be 'NA' if annotateClouds was run in 'sim' mode.

The fifth column contains information about the SSR motif families present in the locus, provided as a means for a user to evaluate how strongly a motif is present in the locus, but this information is not used by the program. If more than one motif family is present at the locus, information for each family is separated by a double colon ('::') in the following format: motif, motif score[3], mean score of family oligos within the locus ((for example, if a family has 3 oligos within a locus with stringencies 2, 4, and 5, then this value = (2+4+5)/3 = 3.67), percentage of the locus that is covered by oligos from the family.

The sixth column contains '+/-', and is only included to be compliant with standard .bed file formatting, representing the strand where the family was found. SSRFinder does not currently search reverse complements of sequences (though it can search for reverse complements of provided motifs), so true value of this depends on whether the sequence provided in the fasta corresponds to the positive or negative strand.

The seventh column shows the length of the locus that is covered by each stringency from any motif family, with lengths for different stringencies separated by a double colon ('::').

exportFPR

necessary command line args: runmode, dir, out, bed, fasta

other args (modified in the control file):

Description: Uses a bed file (normally from an annotated simulated genome) to give false positive rates of different locus lengths and stringencies. Rates represent the percentage of the simulated genome that is covered by each locus and length.

---

[3] Motif score is calculated by assigning each position of the locus that is covered by an oligo from the family a value based on the stringency of the oligo, and then dividing that value by the total length of bases covered by that family in the locus. For example, let's consider a locus that was identified using 5-mer oligos (for the sake of simplicity, in practice we recommend longer oligos). The identified locus is 12 bp in length total (again, this short length is only used as an illustration) and three oligos from a given family are present in the locus. The sequence of the locus is TAAGAGACTTAA, and the oligos for a motif family in the locus (with respective stringencies in parentheses) are: TAAGA(4), AGACT(3), and CTTAA(5). We first assign each position of the locus a value based on the highest stringency oligo present at each position- the first four positions (TAAG) are only covered by the oligo with stringency=4 so the first four values are all 4's; the next position (A) is covered by oligos at stringency 3 and 4 so the next value is a 3; the next two positions (GA) are only covered by an oligo with stringency=3, so the next two values are 2's; the next two positions (CT) are covered by an oligo with stringency=3 and one with stringency=5, so we select the best stringency (3) for the next two values; finally, the remaining three positions (TAA) are only covered by the oligo with stringency=5, so we assign those positions as 5's. The resulting assignments would be 4-4-4-4-3-3-3-3-3-5-5-5. The motif score is the sum of the values at each position (46) divided by the total length covered by oligos from the family (12) = 46/12 = 3.83.

After SSR Finder:

Bed files generated with SSR Finder are compatible with other tools such as bedtools. If desired, a false discovery rate for each locus can be calculated using get_fdr.pl, which can be found at evolutionarygenomics.com. To use get_fdr.pl, provide a bedfile with false positive rates created using the annotateClouds run mode with mode hg and the fasta used to generate the bed file.

Example pipeline:

Below, find an example pipeline that uses test data distributed with the software. Output for each mode is also provided with the software distribution.

Runmode: annotatePerfects

```
perl SSR_finder_v3.52.pl --runmode annotatePerfects --dir . --fasta
hg38_chr22.fa
```

Runmode: buildClouds

```
perl SSR_finder_v3.52.pl --runmode buildClouds --dir . --fasta
hg38_chr22.maskTEs_wout_SSRs.fa --genome hg38.genome
```

Runmode: combineClouds

```
perl SSR_finder_v3.52.pl --runmode combineClouds --dir . --out
testClouds_hg38_chr22_AAG_AATC
```

Runmode: annotateClouds, on simulated genome in mode sim

```
perl SSR_finder_v3.52.pl --runmode annotateClouds --mode sim --fasta
hg38_chr22.fa --assign testClouds_hg38_chr22_AAG_AATC --dir . --
FPTable none –out testClouds_hg38_chr22_AAG_AATC_sim
```

Runmode: exportFPR

```
perl SSR_finder_v3.52.pl --runmode exportFPR --mode sim --fasta
hg38_sim.fa --dir . --bed testClouds_hg38_chr22_AAG_AATC_sim --out
testClouds_hg38_chr22_AAG_AATC_sim_FPR
```

Runmode: annotateClouds on real genome in mode hg

```
perl SSR_finder_v3.52.pl --runmode annotateClouds --mode hg --fasta
hg38_chr22.fa --assign testClouds_hg38_chr22_AAG_AATC --dir . --
FPTable testClouds_hg38_chr22_AAG_AATC_sim_FPR --out
testClouds_hg38_chr22_AAG_AATC_hg
```

To get false discovery rates for each locus:

```
perl get_fdr.pl testClouds_hg38_chr22_AAG_AATC_hg hg38_chr22.fa
```

False discovery rates will be printed in the fourth column of the resulting .bed file.