**Documentation of SSRseq scripts automatizing sequence-based microsatellite genotyping (haplotype approach) though FDSTools.**

1. **General overview of the SSRseq data analysis**

Two *bash* scripts are provided to automated sequence-base microsatellite genotyping from data sequences generated by amplicon sequencing (SSRseq, SSR-GBS or similar approaches as described in Lepais et al. 2019). The first script is used to compare genotype quality generated using different analyses strategies and parameters (steps 1 and 2). The second script is used to generate the final genotypic dataset using optimized parameters (step 3). The general workflow described in full in this documentation is the following:

Step1: FullLenght analysis

* then extract flanking region sequences to run:

Step2: RepeatFocused analysis

* compare results and identify optimized parameters to run:

Step3: Final genotyping

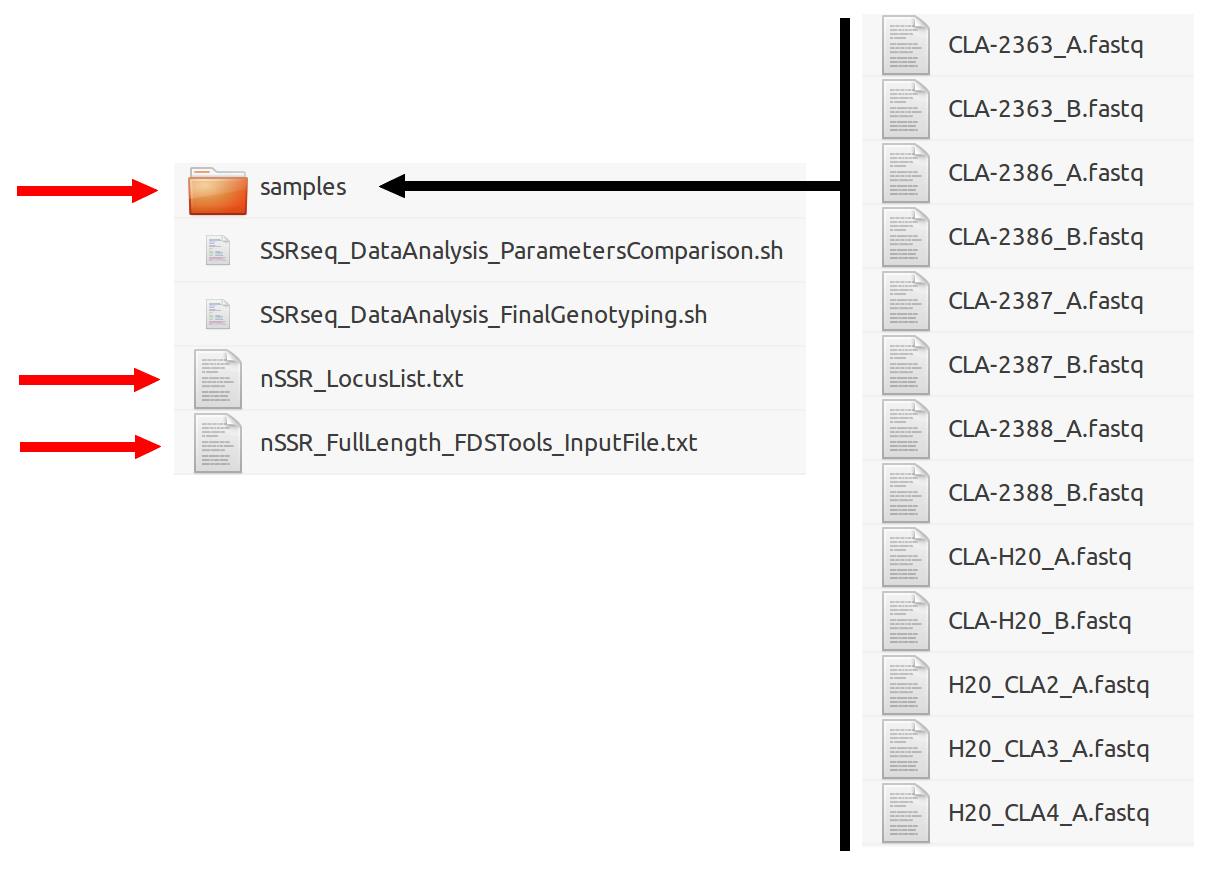
1. **Necessary tools and software**

* Standard computer running on Linux system (tested on Ubuntu); or virtual machine (see below).
* bash (tested on GNU bash v 4.3.48(1))
* Python 2 or 3 (tested on Python 2.7)
* FDSTools (<https://git.lumc.nl/jerryhoogenboom/fdstools>; version 1.2.0). Test that it has correctly been install by typing “fdstools –h” in the terminal.

Alternatively, and more conveniently, **all the necessary tools are already installed and ready to use in the virtual machine “SSRseq-VM.ova”** (based on Ubuntu 18.04, admin login: ssrseq, admin password: ssrseq) that can be hosted in by any Operating System (Windows, Mac OS or Linux). You will need to install VirtualBox (<https://www.virtualbox.org/>), execute it and make “File/Import Virtual Machine” to be able to run the virtual machine within you own computing environment. Because of portability issue, the virtual hard drive space is limited (8 Gb of free space), and thus it is advised to run the analysis in a shared folder named “ssrseq” that VirtualBox expects to find in the Windows root directory (C:\ssrseq). This shared directory should be accessible with read and write access by both Windows and Ubuntu, make it an easy way to exchange files and results. Please create this empty folder to be used for subsequent analysis visible at the path “/media/sf\_ssrseq” within Ubuntu in the virtual machine. For non-Windows host or to configure different shard folder, refer to the VirtualBox virtual machine configuration setting and shared folder setup. Finally, an example script to check for proper setup of the virtual machine can be found in “/home/ssrseq/Documents/ToRunToTest.sh”: opening the script with a text editor (by double-clicking on it) and copy-pasting lines of script into a terminal (black icon with the white “>\_” symbol on the right taskbar) will allow to have an idea about how to run the SSRseq scripts.

1. **Folder and files preparations**

Prepare in one folder (for an example, unzip the example folder):



* A folder named “samples” containing one fastq file per individuals (itself containing sequences from the different sequenced loci). For automated genotyping error estimation based on genotype comparison of repeated individuals, end files name with “\_A.fastq” for the first individual occurrence (or if an individual was not repeated) and “\_B.fastq” for the repetition, with otherwise the exact same file name (for instance : “indivNumber1\_A.fastq” and “indivNumber2\_B.fastq”).
* A text file named “nSSR\_LocusList.txt” indicating the name of the locus to analyse, with each locus name in a different line.
* FDSTools input file, filled in with locus information depending on the analytical strategies: in a first step, polymorphism analysis based on the whole sequence (FullLength strategy), and in a second step, only polymorphism within the repeat region analysed (RepeatFocused strategy). See example files for details and specific format (in addition to FDSTools manuals). For use in this pipeline, several locus parameters should be provided (but not all):
  + fields [aliases], [prefix], [suffix], [length\_adjust], [max\_expected\_copies] and [expected\_allele\_length] should be left empty (or only contained comment lines starting with “;”).
  + [repeat] field indicates the locus name, the sequence of the repeated motif and the minimal (e.g. 0) and maximal number of repeat (e.g. 60). If interrupted or compound microsatellite are sequenced, indicate them as needed.
  + [block length] field indicate the size of the motif. In case of interrupted repeat or compound repeat, only indicate one number as this information is not used in this pipeline, but is needed for the analysis to work.
  + [flanks] field should contain the locus name followed by “ = “, the forward primer sequence, followed by “,” and the reverse-complement reverse primer sequence (using a tool such as <http://arep.med.harvard.edu/labgc/adnan/projects/Utilities/revcomp.html>). See example file for formatting details. Primer sequences are used for the FullLength analysis strategy as the sequence between the primer are inspected for polymorphism analysis. For the RepeatFocused analysis, primer sequence and flanking sequence should be indicated (in details, the concatenation of the forward primer sequence and the flanking region sequence to the left of the microsatellite, and the flanking sequence to the right of the microsatellite and the reverse-complement of the reverse sequence). The easiest way to extract these sequences is to use previous results from the FullLength analysis strategy. FDSTools input fiels have to be named “nSSR\_FullLength\_FDSTools\_InputFile.txt” and “nSSR\_RepeatFocused\_FDSTools\_InputFile.txt”.

1. **Pipeline execution step 1: comparing analysis parameters with full sequence length approach**

* Open a shell terminal and go (“cd”) to the working directory (containing the scripts, the input files and the folder with sequences)
* Make scripts executable by typing in a terminal:

chmod +x SSRseq\_DataAnalysis\_ParametersComparison.sh

chmod +x SSRseq\_DataAnalysis\_FinalGenotyping.sh

* then execute the first script by typing:

./SSRseq\_DataAnalysis\_ParametersComparison.sh nSSR FullLength ParameterSet1 50 10 15 20 ParameterSet2 70 10 10 20 50 10 2

will execute the whole data analysis, analysing sequence polymorphism between primers, and for two parameter sets (arbitrary called “ParameterSet1” and “ParameterSet2”) of Stuttermarks and Allelefinder. The parameter value indicated in the command line above correspond to the pipeline default values, so that the exact same result can be obtained by simply typing:

./SSRseq\_DataAnalysis\_ParametersComparison.sh

It is still preferable to indicate them in full to keep record track of the parameter used. Although these two parameter sets have been showed to give good results for various loci, different parameter combination could be tested by changing the parameter value. Their effect on the analysis algorithm are thus explained below, and additional clarification can be founded in the original FDSTools paper (Hoogenboom et al. 2017; <https://doi.org/10.1016/j.fsigen.2016.11.007>) and in FDSTools manuals accessed by typing “fdstools –h”, “fdstools tssv –h”, “fdstools stuttermark –h” or “fdstools allelefinder –h” in a terminal.

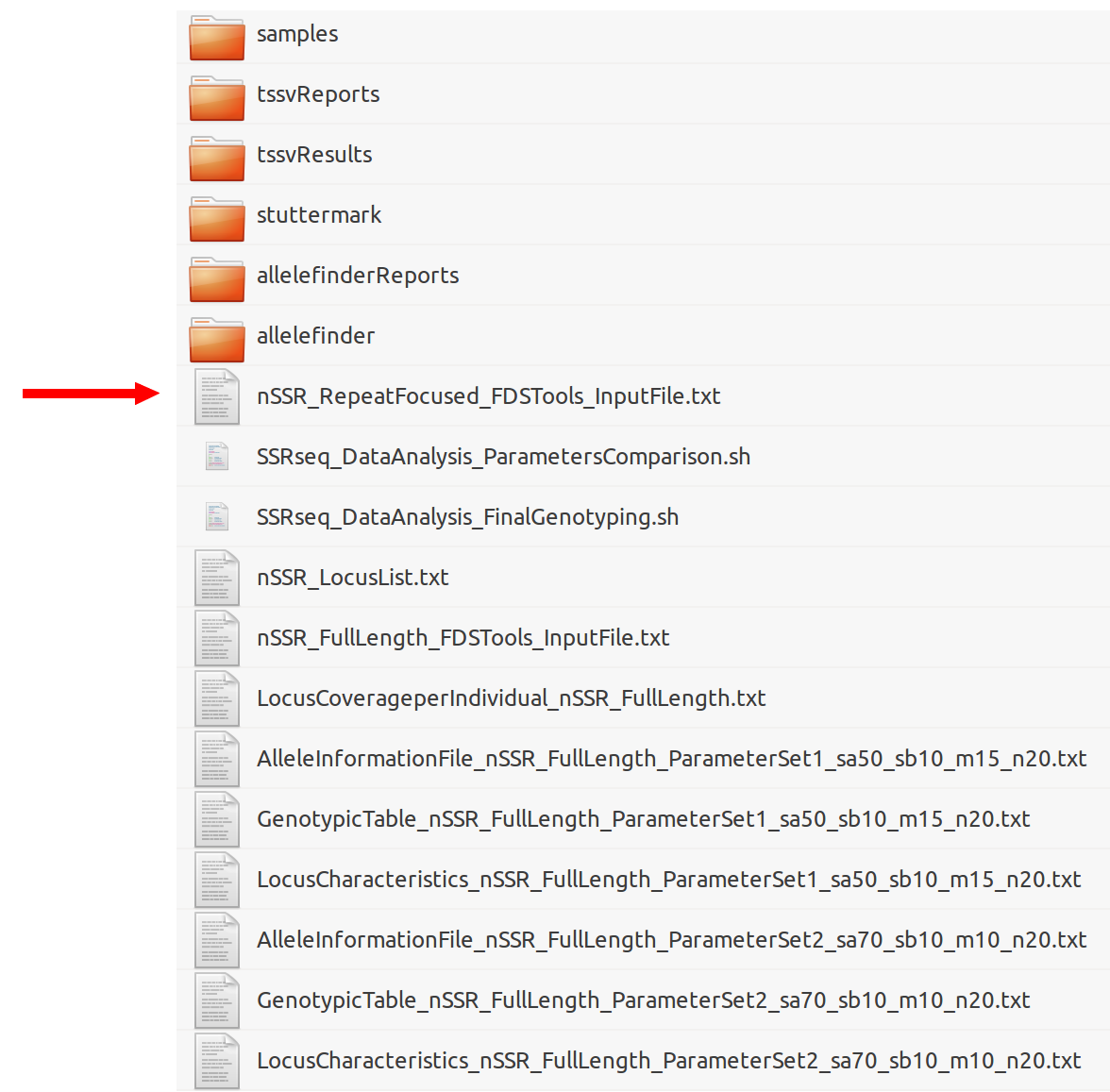
**Details of FDSTools analysis parameters**. For Stuttermark, a sequence with a loss or a gain of one repeat compared to another sequence but with high coverage will not be flagged as potential stutter, depending on user defined thresholds controlling for the relative coverage of the two sequences (parameter –s). For ParameterSet1, we assumed that a sequence in stutter configuration with -1 repeat is flagged as stutter if its coverage is below 50% of the originating sequence (this corresponds to the first parameter after “ParameterSet1” in the command line) and stutter configuration with +1 repeat is flagged as stutter if its coverage is below 10% of the originating sequence (this corresponds to the second parameter after “ParameterSet1” in the command line). For ParameterSet2, we assumed that a sequence in stutter configuration with -1 repeat is flagged as stutter if its coverage is below 70% of the originating sequence (this corresponds to the first parameter after “ParameterSet2” in the command line) and stutter configuration with +1 repeat is flagged as stutter if its coverage is below 10% of the originating sequence (this corresponds to the second parameter after “ParameterSet2” in the command line). For -2 stutter sequence, the rule is set to 50% and 70% for ParameterSet1 and ParameterSet2 respectively when comparing -2 and -1 repeated sequence. Allelefinder ignores sequences flagged as noise or stutter and call one or two alleles from all potential allele based on user defined coverage thresholds. For ParameterSet1, a heterozygote genotype is called if the second allele has at least 15% of coverage compared to the highest covered allele for a locus (parameter –m 15), this corresponds to the third parameter after “ParameterSet1” in the command line. For ParameterSet2, this parameter was set to 10% (-m 10), this corresponds to the third parameter after “ParameterSet2” in the command line. ParameterSet1 and ParameterSet2 shared some parameter settings: we required a minimum of 20 reads for the most covered allele for a locus in an individual (parameter –n 20), otherwise a missing genotype is called (this corresponds to the third parameter after “ParameterSet1” and “ParameterSet2” in the command line). In addition, for Allelefinder, specific parameter controlling for potential sample contamination can be adjust: a locus is called missing data if the most covered non-allelic sequence (i.e. additional allele beyond the ploidy level setup by the last parameter, here 2) is at least 50% of the most covered allele (parameter –M 50, the ante-penultimate parameter) and a sample is rejected if more than 10 loci have a high number of non-allelic sequences (parameter –x 10, the penultimate parameter). These parameters can be adjusted when DNA contaminations or paralogous loci are suspected, for instance by decreasing the ante-penultimate parameter to 33. Locus with high coverage but high missing data probably amplify paralogs, and samples with high sequence coverage but high missing genotypes are probably contaminated.

**Outputted files.** Running the above command line will produce a file:

* “LocusCoverageperIndividual\_nSSR\_FullLength.txt“ reporting coverage for each locus (in row) and individuals (in column)

and two sets of output files (one for each ParameterSet):

* “AlleleInformationFile\_nSSR \_FullLength\_[…].txt”: one line per detected allele (haplotype) providing the locus name, the annotated allele sequence (with the number of repeated indicate under bracket), the allele code corresponding to the genotypic notation in the genotypic table, the number of occurance across the dataset, the full allele sequence, and its length
* “GenotypicTable\_nSSR \_[…].txt”: the genotype of each individual (in row) at each locuss (in column) ready to copy and paste into a spreadsheet. Missing genotypes are coded as “NA”. Alleles are coded as three digits string for compatibility with standard population genetic software (see AlleleInformationFile for correspondence between the code and the allele sequence).
* “LocusCharacteristics\_nSSR \_[…].txt”: indicates for each locus, the number of haplotypes (allele differing in sequence), the number of alleles based on size, the percentage of missing data, the number of allelic mismatch and the number of individual compared, the percentage of allelic error. The format allows for direct copy and paste into a spreadsheet for further analysis (for instance, comparison between different parameter sets).

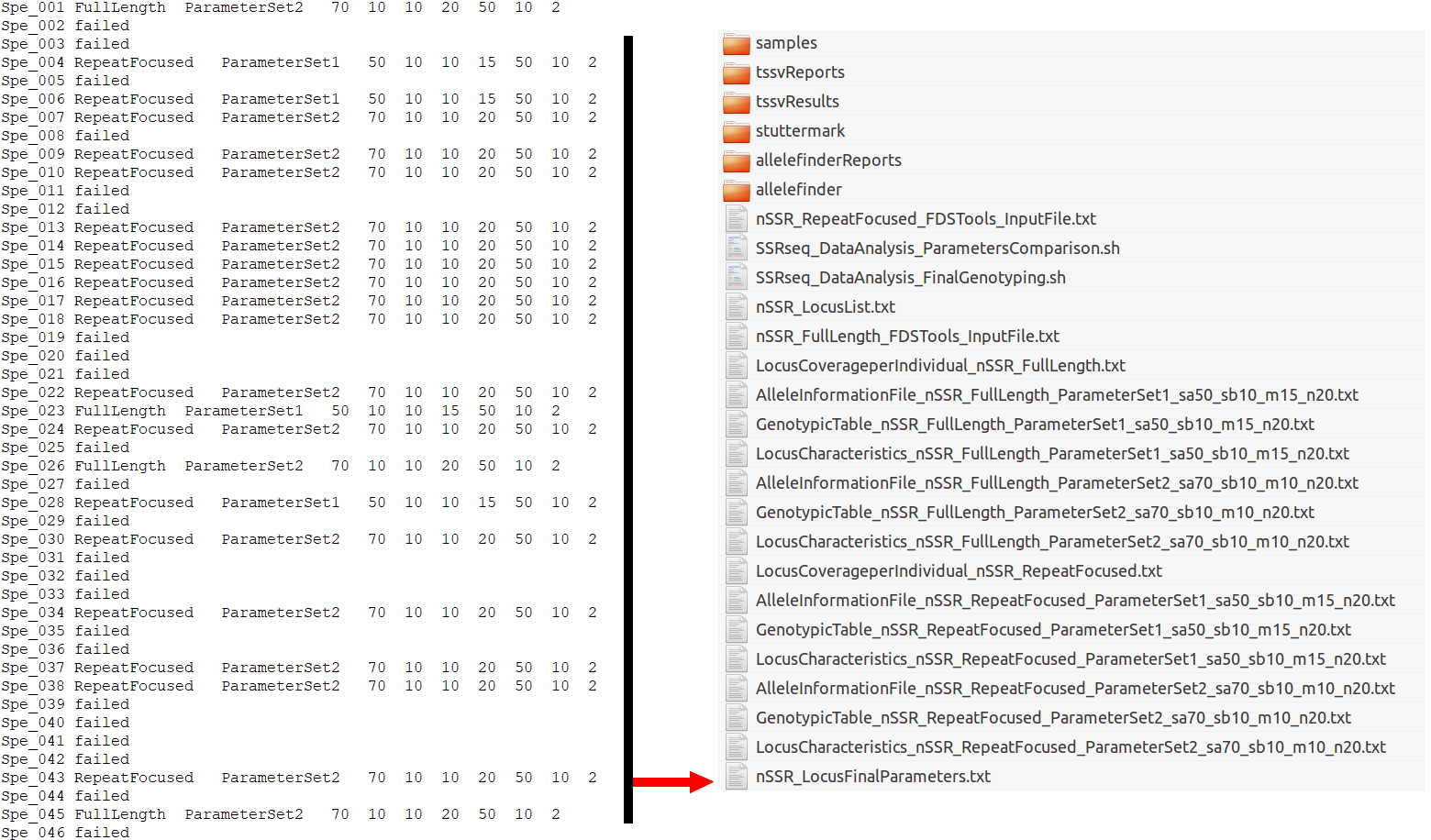


1. **Pipeline execution step 2: comparing analysis parameters with repeat focused approach**

Then, the AlleleInformationFile can be used to retrieve flanking sequences around repeated motifs (the “AlleleSequenceAnnotated” field is useful for this purpose) and used to inform FDSTools input field for the RepeatFocus analysis strategy. Once ready, it can be run by typing:

./SSRseq\_DataAnalysis\_ParametersComparison.sh nSSR RepeatFocused ParameterSet1 50 10 15 20 ParameterSet2 70 10 10 20 50 10 2

Note that additional output files will be produce without replacing previous results. Then, the best analysis strategy and parameter set can be determined for each locus based on allelic error, missing data and number of alleles.



1. **Pipeline execution step 3: final genotyping using optimized parameter for each locus**

Once the best analysis strategy and parameter has been determined for each locus, a final and optimized analysis can automatically be performed based in the information provided by a file named “nSSR\_LocusFinalParameters.txt” which contains for each locus, its name, the analysis strategy (“RepeatFocused”, “FullLength”, or “failed” is the locus should not be part of the final dataset), the parameter set to apply, followed by the corresponding parameters (as in the default setting or see above for the effect of each the three settable parameters), separated by tabulation (see example the file for details). Be careful to transform any Windows end of line character into Linux end of line (either using the Notepad++ conversion, or typing:

sed 's/^M$//' LocusFinalParameters.txt

into the terminal. Then, the final genotyping is performed by typing in the terminal:

./SSRseq\_DataAnalysis\_FinalGenotyping.sh nSSR

note that there are no parameters as they are pass thought the “LocusFinalParameters.txt” file. This script will automatically create and format FDStools input files, performed the required analyses for each locus and compile all results into three output files: locus and allele information files and the final genotypic table (all as described above).

