ddRADseqTools 0.43

Software package for *in silico* simulation and testing of double digest RADseq experiments

GI Genética, Fisiología e Historia Forestal
Dpto. Sistemas y Recursos Naturales
ETSI Montes, Forestal y del Medio Natural
Universidad Politécnica de Madrid

http://gfhforestal.com/
https://github.com/ggfhf/

Table of contents

Introduction

ddRADseqTools is a set of applications useful to *in silico* design and testing of double digest RADseq (ddRADseq) experiments. Briefly, ddRADseq is a method of genotyping that first digests a genome with a pair of restriction endonucleases, and then uses high-throughput sequencing technologies (Illumina) to obtain sequences from these fragments. The technique was developed by (Peterson *et al.* 2012), and allows to genotype high number of individuals by obtaining polymorphisms (mainly SNPs but also indels) across the whole genome.

This software package is indicated to simulate ddRADseq read files under varying scenarios. When a reference genome is available, the program *rsitesearch.py* extracts the fragments resulting from a digestion with a particular pair of restriction endonucleases. When there is not a reference available, the program *fragsgeneration.py* generates random fragments. Once the fragments have been obtained, the program *simddradseq.py* simulates PE or SE Illumina ddRADseq raw read files. This program allows handling a wide number of parameters, such as modifications of ddRADseq library construction, read types (SE or PE), number of reads, size of the fragments, mutation (substitutions or indels) probability, or PCR duplicates probability.

ddRADseqTools contains four additional tools: 1) a demultiplexer, indsdemultiplexing.py; 2) a PCR duplicate removal/quantification tool, pcrdupremoval.py; 4) a trimmer, readstrim.py, that cuts the adapters, primer, indexes and DBR from raw reads; and 5) a sequence locator, to assess the position of a particular nucleotide sequence in a reference genome, seqlocation.py.

The output of these applications are ready to be submitted to alignment utilities, such as BWA, or to (dd)RADseq analysis pipelines, such as STACKS or Pyrad.

A flow chart of the programs contained in ddRADseqTools is shown in Figure 1.

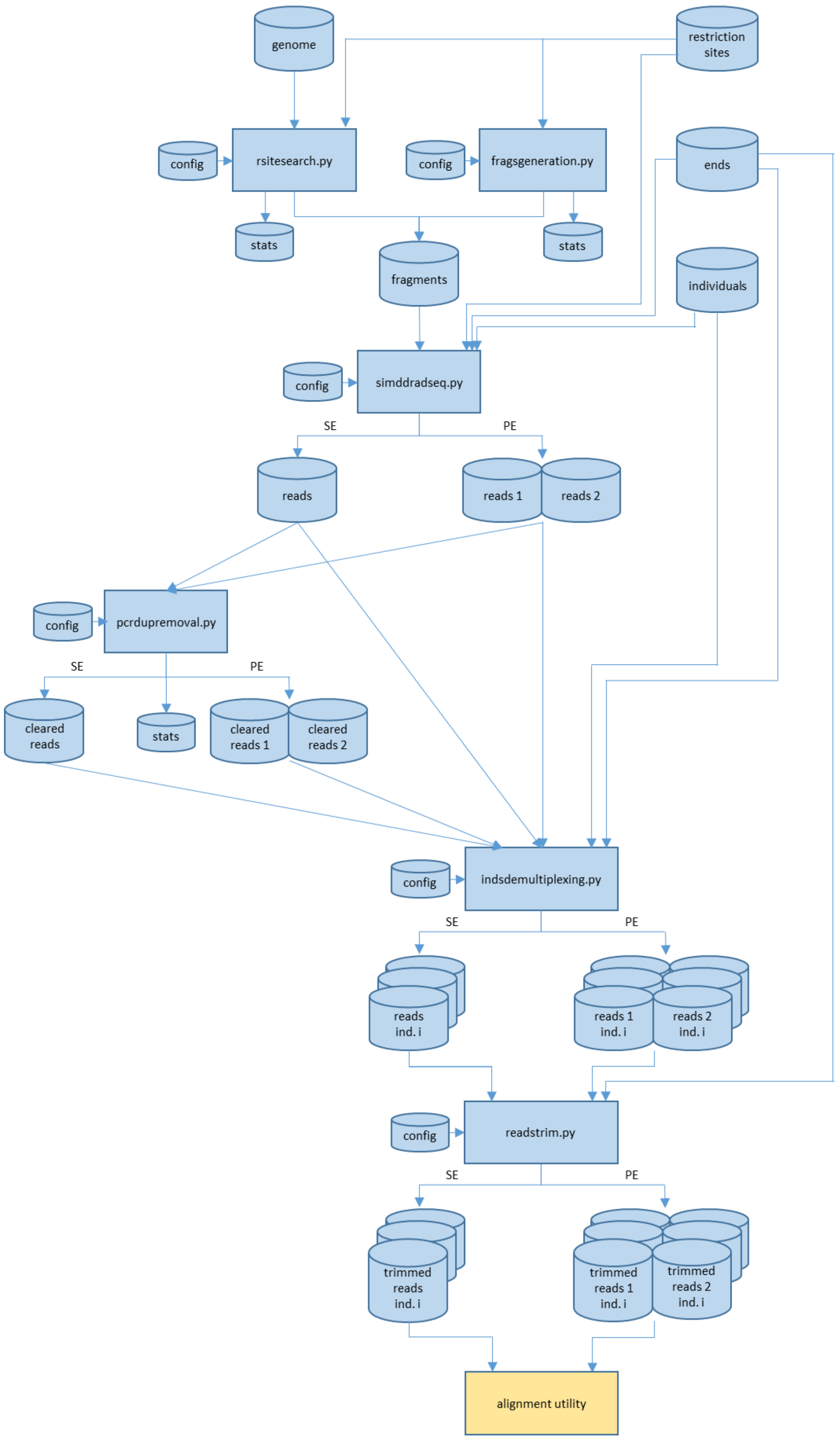

Figure. 1. Flow-chart of ddRADseqTools

Installation

ddRADseqTools was programmed in Python3, and runs in any computer with an OS that allows for Python 3: Linux/Unix, Microsoft Windows, Mac OS X and other platforms. If Linux is the OS of your computer, Python will be already installed. In other cases, it can be downloaded from https://www.python.org/.

The only dependencies required to run this software package are the NumPy and matplotlib libraries (see the installation instructions for these libraries in http://www.numpy.org/ and http://matplotlib.org/, respectively).

Several distributions include both Python and other packages among which are Numpy and Matplotlib. One of such distributions is Anaconda, a free cross-platform for Linux, Windows and Mac OS X. (https://store.continuum.io/cshop/anaconda/).

ddRADseqTools is available from the GitHub software repository (https://github.com/GGFHF/ddRADseqTools), and is distributed under GNU General Public Licence Version 3.

To install ddRADseqTools in Linux, simply download and decompress the *ddRADseqTools*.zip into a directory, typing the following command in a terminal window:

```
$ unzip ddRADseqToolsTools.zip
```

Execution permissions of the programs have to be set by using this command:

```
$ chmod u+x *.py
```

Examples of command line instructions to run the programs:

```
$ ./rsitesearch.py  --genfile=genome_Ptaeda.fasta --fragsfile=frags_Ptaeda.fasta
```

```
$ ./simddradseq.py --fragsfile=frags_Ptaeda.fasta --readtype=PE
```

Instructions for installation in other OS are equivalent to those for Linux.

Package files

The software package ddRADseqTools has the following files:

- *ddRADseqTools-manual.pdf*: The manual that describes the operation of the software package.
- *ends.txt*: It contains the sequence ends of the reads to be simulated. The sequence ends are integrated by the sequence of a primer followed by the sequence of an adaptor. It is possible to modify this file by adding new end sequences at the user's choice.
- *fragsgeneration.py*: This program generates random fragments emulating the digestion of a genome with two particular restriction endonucleases.
- *fragsgeneration-config.txt*: It contains the configuration options to run *fragsgeneration.py*.
- *genlib.py*: It contains the general functions and classes used by the programs of ddRADseqTools.
- *Individuals*.txt*: Several example files. Each file contains sequences identifying each individual of the experiment: its index sequence in the *Adapter 1* and, optionally, its index sequence in the *Adapter 2*. It is necessary to update the data in every experiment if the index sequences are modified.
- *indsdemultiplexing.py*: This program demultiplexes one file (SE) or two files (PE) with reads of *n* individuals in *n* files (SE) or *2n* files (PE) that contain the reads of each individual.
- *indsdemultiplexing-config.txt*: It contains the configuration options to run *indsdemultiplexing.py.*
- *pcrdupremoval.py*: This program allows to quantify PCR duplicates, to calculate the percentage of missing data by individual and locus and to remove them from one file (SE) or two files (PE) in FASTQ/FASTA files from a ddRADseq experiment.
- *pcrdupremoval-config.txt*: It contains the configuration options to run *pcrdupremoval.py.*
- *readstrim.py*: This program trims the ends of one read file (SE) or two read files (PE), i. e. it cuts the adapters and other Illumina sequences.
- *readstrim-config.txt*: It contains the configuration options to run *readstrim.py.*
- *restrictionsites.txt*: It contains restriction sites sequences of restriction endonucleases of common use, and the motifs corresponding to their cut sites. This file can be modified to add new restriction enzymes.
- *rsitesearch.py*: This program extracts the fragments resulting from an *in silico* digestion of a reference genome with two particular restriction endonucleases.
- *rsitesearch-config.txt*: It contains the configuration options to run *rsitesearch.py*.
- *seqlocation.py*: This program finds the position of a particular nucleotide sequence in a reference genome.
- *seqlocation-config.txt*: It contains the configuration options to run *seqlocation.py.*
- *simddradseq.py*: This program builds files in FASTQ/FASTA format with Illumina SE or PE simulated reads from a virtual library of a ddRADseq experiment.
- *simddradseq-config.txt*: It contains the configuration option to run *simddradseq.py.*
- *simulation-ddradseq.sh*: This script simulates reads of a ddRADseq obtained from fragments previously obtained of the digest of the genome of *S. cerevisiae*, *H. sapiens*

and *P. taeda* varying two options: number of reads to generate and probability of loci bearing PCR duplicates.

- *simulation-dropout.sh*: This script analyses the effect of dropout on the number of reads to be generate and the probability of loci bearing PCR duplicates.
- *simulation-gcfactor.sh*: This script analyses the effect of GC factor on the number of reads to be generate and the probability of loci bearing PCR duplicates.
- *simultation-genome.sh*: This script analyses how several enzymes pairs perform a double digest of the genome of *S. cerevisiae*, *H. sapiens* and *P. taeda*.
- *simulation-mutations.sh*: This script extracts statitics of mutated and non-mutated reads.
- *simulation-pcrdupprob.sh*: This script analyses the effect of the probability of PCR duplicates on the number of reads to generate.
- *simulation-performance.sh*: This script studies the performance of the ddRADseqTols programs.
- *simulation-pipeline*.sh*: Several example scripts. Each script generates fragments from a genome, simulates a double digest, generates their reads (SE or PE), removes PCR duplicates (if it is applicable), demultiplexes the individuals, trims the adapters and other Illumina specific sequences, aligns the reads and gets SAM, BAM, BED and VCF format files to study and visualize alignments.
- *simulation-poissonparam.sh*: This script analyses the effect of parameter lambda of Poisson distribution in the calculation of the PCR duplicates number of each locus of each individual.
- *simulation-test-pe-fasta.bat*: This script is similar to *simulation-test-pe-fasta.sh* but it is designed for Windows OS.
- *simulation-test-pe-fasta.sh*: This script executes a test of each program of the software package ddRADseqTools to PE reads with FASTA format.
- *simulation-test-pe-fastq.bat*: This script is similar to *simulation-test-pe-fastq.sh* but it is designed for Windows OS.
- *simulation-test-pe-fastq.sh*: This script executes a test of each program of the software package ddRADseqTools to PE reads with FASTQ format.
- *simulation-test-se-fasta.bat*: This script is similar to *simulation-test-se-fasta.sh* but it is designed for Windows OS.
- *simulation-test-se-fasta.sh*: This script executes a test of each program of the software package ddRADseqTools to SE reads with FASTA format.
- *simulation-test-se-fastq.bat*: This script is similar to *simulation-test-se-fastq.sh* but it is designed for Windows OS.
- *simulation-test-se-fastq.sh*: This script executes a test of each program of the software package ddRADseqTools to SE reads with FASTQ format.
- *simulation-unequal-coverage.sh*: This script analyses the coverage variation among individuals across loci.

rsitesearch.py

This program locates the restriction sites motifs and performs an *in silico* double digestion of a genome. The output is a FASTA file with the resulting fragments. It also provides the GC rate distribution of the fragments and some statistics regarding the number of fragments classified according to fragment size intervals.

The input and output files of *rsitesearch.py* are shown in Figure 1 as well as the position of this program within the processes flow of ddRADseqTools.

The options of the program are detailed in Table 1.

Table 1. *rsitesearch.py* options.

Option	Default value	Comment
genfile	./genome.fna	Path of the reference genome file in FASTA format or .gz format (compressed).
fragsfile	./fragments.fasta	Path of the output fragments file.
rsfile	./restrictionsites.txt	Path of the input restriction sites file.
enzyme1	EcoRI	Name of the first restriction enzyme used in rsfile. Instead of the name, the restriction site sequence is allowed. e. g. EcoRI and GAATTC are equivalent.
enzyme2	MseI	Name of the second restriction enzyme used in rsfile. Instead of the name, the restriction site sequence is allowed. e. g. MseI and TTAA are equivalent.
minfragsize*	201	Lower fragment loci size.
maxfragsize*	300	Upper fragment loci size.
fragstfile	./genome-statistics.txt	Output statistics file.
fragstinterval	25	Interval length of fragment size for the output statistics.
plot	YES	Statistical graphs: YES or NO.
verbose	YES	Additional job status info during the run: YES or NO.
trace	NO	Additional info useful to the developer team: YES or NO.

*During library construction in ddRADseq experiments it is very common to filter only the fragments ranging a particular size (usually 100-400 bp). This size interval can be set here.

fragsgeneration.py

This program generates random fragments simulating a double digestion of a genome and writes them in a FASTA file. It is useful when *rsitesearch.py* cannot be used because there is not reference genome. It also provides the GC rate distribution of the fragments and some statistics regarding the number of fragments classified according to fragment size intervals.

The input and output files of *fragsgeneration.py* are shown in Figure 1 as well as the position of this program within the processes flow of ddRADseqTools.

The options of the program are detailed in Table 2.

Table 2. *fragsgeneration.py* options.

Option	Default value	Comment
fragsfile	./fragments.fasta	Path of the output fragments file.
rsfile	./restrictionsites.txt	Path of the input restriction sites file.
enzyme1	EcoRI	Name of the first restriction enzyme used in rsfile. Instead of the name, the restriction site sequence is allowed. e. g. EcoRI and GAATTC are equivalent.
enzyme2	MseI	Name of the second restriction enzyme used in rsfile. Instead of the name, the restriction site sequence is allowed. e. g. MseI and TTAA are equivalent.
fragsnum	10000	Number of fragments to generate.
minfragsize*	201	Lower fragment loci size.
maxfragsize*	300	Upper fragment loci size.
fragstfile	./genome-statistics.txt	Output statistics file.
fragstinterval	25	Interval length of fragment size for the output statistics.
plot	YES	Statistical graphs: YES or NO.
verbose	YES	Additional job status info during the run: YES or NO.
trace	NO	Additional info useful to the developer team: YES or NO.

*During library construction in ddRADseq experiments it is very common to filter only the fragments ranging a particular size (usually 100-400 bp). This size interval can be set here.

simddradseq.py

This program builds Illumina SE or PE simulated read files from a virtual library of a ddRADseq experiment in FASTQ/FASTA format.

The input fragments can be obtained in two ways:

1) Using a reference genome via *rsitesearch.py*.
2) Randomly via *fragsgeneration.py*.

Several modifications of the ddRADseq library construction methodology of (Peterson *et al.* 2012) exist. In this version of ddRADseqTools we have implemented three of these techniques:

1) *IND1*: It is the original ddRADseq methology by (Peterson *et al.* 2012). An index sequence is inserted in the adapter 1.
2) *IND1_DBR*: An index sequence and a degenerate base region (DBR) to allow PCR duplicates identification are inserted in the adapter 1 (Casbon *et al.* 2011; Schweyen *et al.* 2014; Tin *et al.* 2015).
3) *IND1_IND2*: In addition to the index sequence in adapter 1, another index sequence is inserted in the adapter 2 (Peterson *et al.* 2012; Mastretta-Yanes *et al.* 2015).
4) *IND1_IND2_DBR*: This technique uses two index sequences, one in adapter 1 and another in adapter 2, and a DBR (Casbon *et al.* 2011; Schweyen *et al.* 2014; Tin *et al.* 2015).

This program generates SE or PE reads of user's defined length in FASTQ or FASTA format.

The coverage is controlled by setting the number of fragment loci, the number of individuals (controlled by the configuration file *individuals.txt*) and the number of reads of the library. Coverage may be unequal among loci and individuals, ranging between the given values for *minreadvar* and *maxreadvar*. If uniform coverage is desired, these options should be set to 1.

Several options can be set to simulate mutation between individuals by including a probability of mutation, the maximum mutations number by locus and a probability of indel of varying size occurrence. In the current version, only the Jukes-Cantor model of evolution is implemented, and phylogenetic relationships between individuals or groups of individuals cannot be simulated. If *BC_IND_DBR* is used, the probability of having PCR duplicates in a locus is incorporated.

Also the simulation of technical replicates is allowed (see in the file *individuals.txt*).

The input and output files of *simddradseq.py* are shown in Figure 1 as well as the position of this program within the processes flow of ddRADseqTools.

The options of the program are detailed in Table 3.

Table 3 *simddradseq.py* options.

Option	Default value	Comment
fragsfile	./fragments.fasta	Path of the input fragments file.

technique	IND1_IND2_DBR	Three methodologies are available: IND1 (an index sequence in adapter 1), IND1_DBR (an index sequence + a DBR in adapter 1), IND1_IND2 (an index sequence in adapter 1 + an index sequence in adapter 2) and IND1_IND2_DBR (an index sequence in adapter 1 + an index sequence in adapter 2 + a DBR).
format	FASTQ	Format of the output file: FASTQ or FASTA.
readsfile	./reads	Path of the output read file (without extension).
readtype	PE	Read type: SE or PE.
rsfile	./restrictionsites.txt	Path of the input restriction sites file.
enzyme1	EcoRI	Name of the first restriction enzyme used in rsfile. Instead of the name, the restriction site sequence is allowed. e. g. EcoRI and GAATTC are equivalent.
enzyme2	MseI	Name of the second restriction enzyme used in rsfile. Instead of the name, the restriction site sequence is allowed. e. g. MseI and TTAA are equivalent.
endsfile	./ends.txt	Path of the input end sequences file.
index1len	6	Index sequence length in Adapter 1.
index2len	6	Index sequence length in Adapter 2 (it must be 0 when the technique is BC).
dbrlen	4	DBR length (it must be 0 when the technique is BC or BC_IND).
wend	end01	Code used in endsfile corresponding to the end where adapter 1 is.
cend	end02	Code used in endsfile corresponding to the end where adapter 2 is.
individualsfile	./individuals.txt	Path of the input individuals file.
locinum	100	Number of loci to sample.
readsnum	10000	Number of reads to generate.
minreadvar	0.8	Lower parameter value of the interval to control variation of the number of reads per locus (0.5 <= minreadvar<= 1.0).
maxreadvar	1.2	Upper parameter value of the interval to control variation of the number of reads per locus (1.0 <= maxreadvar <= 1.5).
insertlen	180	Insert length, i. e. genome sequence length inserted in the reads.
mutprob	0.2	Mutation probability (0.0 <= mutprob < 1.0.)
locusmaxmut	1	Maximum mutation number by locus (1 <= locusmaxmut <= 5)
indelprob	0.4	Indel probability (0.0 <= indelprob < 1.0). This is the probability of a mutation being an indel (otherwise, it will be a substitution).
maxindelsize	3	Maximum size of the generated indels (1 <= maxindelsize < 20).
dropout	0	Probability of mutation at the enzyme recognition sites (0.0 <= dropout < 1.0).
pcrdupprob	0	Probability of loci bearing PCR duplicates (0.0 <= pcrdupprob < 1.0).
pcrdistribution	MULTINOMIAL	Distribution type to calculate the PCR duplicates number: MULTINOMIAL or POISSON

multiparam	0.333,0.267,0.200,0.133,0.067	Probability values to multinomial distribution with format prob1,prob2,...,probn (they must sum 1.0)
poissonparam	1.0	Lambda value of the Poisson distribution
gcfactor	0	Weight factor of GC ratio in a locus with PCR duplicates (0.0 <= gcfactor < 1.0)
verbose	YES	Additional job status info during the run: YES or NO.
trace	NO	Additional info useful to the developer team: YES or NO.

pcrdupremoval.py

This program quantifies and removes the PCR duplicates obtained from a ddRADseq experiment that uses a degenerate base region (DBR) embedded in the adapters in addition to the index sequences (Schweyen *et al.* 2014; Tin *et al.* 2015). Also it calculates the percentage of missing data by individual and locus

The input reads have been generated by *simddradseq.py*.

The input and output files of *pcrdupremoval.py* are shown in Figure 1 as well as the position of this program within the processes flow of ddRADseqTools.

The options of the program are detailed in Table 4.

Table 4. *pcrdupremoval.py* options.

Option	Default value	Comment
format	FASTQ	Format of the output file: FASTQ or FASTA.
readtype	PE	Read type: SE or PE.
readsfile1	./reads_1.fastq	Path of the file for SE reads or the reads file where *Adapter 1* is for PE reads.
readsfile2	./reads_2.fastq	Path of reads file where *Adapter 2* is for PE reads or NONE for SE reads.
clearfile	./reads_cleared	Path of the output file with removed PCR duplicates (without extension).
dupstfile	./pcrduplicates_stats.txt	Path of the PCR duplicates statistics file.
plot	YES	Statistical graphs: YES or NO.
verbose	YES	Additional job status info during the run: YES or NO.
trace	NO	Additional info useful to the developer team: YES or NO.

indsdemultiplexing.py

This program demultiplexes one file (SE) or two files (PE) with reads of *n* individuals in *n* files (SE) or 2*n* files (PE), containing the reads of each individual.

The input reads have been generated by *simddradseq.py* or they have been the result of the removal of PCR duplicates performed with *pcrdupremoval.py*.

The input and output files of *indsdemultiplexing.py* are shown in Figure 1as well as the position of this program within the processes flow of ddRADseqTools.

The options of the program are detailed in Table 5.

Table 5. *indsdemutiplexing.py* options.

Option	Default value	Comment
technique	IND1_IND2_DBR	Three methodologies are available: IND1 (an index sequence in *Adapter 1*), IND1_DBR (an index sequence + a DBR in adapter 1), IND1_IND2 (an index sequence in *Adapter 1* + an index sequence in *Adapter 2*) and IND1_IND2_DBR (an index sequence in *Adapter 1* + an index sequence in *Adapter 2* + a DBR).
format	FASTQ	Format of the output file: FASTQ or FASTA.
readtype	PE	Read type: SE or PE.
endsfile	./ends.txt	Path of the input end sequences file.
index1len	6	Index sequence length in *Adapter 1*.
index2len	6	Index sequence length in *Adapter 2* (it must be 0 when technique is BC).
dbrlen	4	DBR length (it must be 0 when technique is IND1 or IND1_IND2).
wend	end01	Code used in endsfile corresponding to the end where *Adapter 1* is.
cend	end02	Code used in endsfile corresponding to the end where *Adapter 2* is.
individualsfile	./individuals.txt	Path of the input individuals file.
readsfile1	./reads_1.fastq	Path of the file for SE reads or the reads file where *Adapter 1* is for PE reads.
readsfile2	./reads_2.fastq	Path of reads file where *Adapter 2* is for PE reads or NONE for SE reads.
verbose	YES	Additional job status info during the run: YES or NO.
trace	NO	Additional info useful to the developer team: YES or NO.

readstrim.py

This program trims the ends of 1 file (SE) / 2 files (PE) of reads, i.e. cuts the adapters and other Illumina sequences.

The input and output files of *readstrim.py* are shown in Figure 1 as well as the position of this program within the processes flow of ddRADseqTools.

The options of the program are detailed in Table 6.

Table 6. *readstrim.py* options.

Option	Default value	Comment
technique	IND1_IND2_DBR	Three methodologies are available: IND1 (an index sequence in *Adapter 1*), IND1_DBR (an index sequence + a DBR in adapter 1), IND1_IND2 (an index sequence in *Adapter 1* + an index sequence in *Adapter 2*) and IND1_IND2_DBR (an index sequence in *Adapter 1* + an index sequence in *Adapter 2* + a DBR).
format	FASTQ	Format of the output file: FASTQ or FASTA.
readtype	PE	Read type: SE or PE.
endsfile	./ends.txt	Path of the input end sequences file.
index1len	6	Index sequence length in *Adapter 1*.
index2len	6	Index sequence length in *Adapter 2* (it must be 0 when technique is BC).
dbrlen	4	DBR length (it must be 0 when technique is IND1 or IND1_IND2).
wend	end01	Code used in endsfile corresponding to the end where *Adapter 1* is.
cend	end02	Code used in endsfile corresponding to the end where *Adapter 2* is.
readsfile1	./reads_1.fastq	Path of the file for SE reads or the reads file where *Adapter 1* is for PE reads.
readsfile2	./reads_2.fastq	Path of reads file where *Adapter 2* is for PE reads or NONE for SE reads.
trimfile	./reads_cleared	Path of the output file with trimmed reads (without extension).
verbose	YES	Additional job status info during the run: YES or NO.
trace	NO	Additional info useful to the developer team: YES or NO.

seqlocation.py

This program locates a sequence into the genome and shows the start and end positions as well as the reverse complementary sequence. No mismatches are allowed in this version of the program.

The options of the program are detailed in Table 7.

Table 7. *seqlocation.py* options.

Option	Default value	Comment
genfile	./genome.fna	File of the reference genome in FASTA format. The file can be compressed.
seq	TGGAGGTGGGG	The sequence to be located into the genome.
verbose	YES	Additional job status info during the run: YES or NO.
trace	NO	Additional info useful to the developer team: YES or NO.

Common characteristics of programs

The programs of ddRADseqTools have two common characteristics: 1) how the options are processed; and 2) how the help is invoked. *rsitesearch.py* is used as an example to illustrate these two characteristics.

The file *rsitesearch-config.txt* allows configuring the options detailed in the previous table. When this file is created, it shows the default parameters. Default options can be changed in two ways:

- By editing the *rsitesearch-config.txt* file in a text editor
- Including parameters in the command line. The program runs as:

```
$ ./rsitesearch.py [--option=<value> [--option=<value>, ...]]
```

That is, the options can be passed with their new values as parameters. Not modified options are read by the program as they appear in *rsitesearch-config.txt*.

If the file *rsitesearch-config.txt* needs to be built again (e. g. the file has been deleted or the file has been incorrectly modified), the next instruction will do it automatically:

```
$ ./rsitesearch.py --config
```

Further information regarding parameters should be consulted in the help of the program, which is shown with the following command:

```
$ ./rsitesearch.py --help
```

For other programs, replacing *rsitesearch.py* and *rsitesearch-config.txt* by the program name and the name of its configuration file.

Example scripts

Scripts can be written to facilitate the execution of the ddRADseqTools programs in non-interactive mode in order to perform repetitive tasks or processing a large number of files.

Eleven examples of Linux Bash scripts and four Windows scripts have been included in the software package ddRADseqTools to show the way the programs can be used. Some of these scripts are:

- *simulation-test-se-fastq.sh*: This script executes a test of each program of the software package ddRADseqTools to SE reads with FASTQ format. There is a windows version of this script attached, *simulation-test-se-fastq.bat*.
- *simulation-genomes.sh*: This script analyses the performance of several enzymes pairs in a double digest of the genomes of *Saccharomyces cerevisiae*, *Homo sapiens* and *Pinus taeda*.
- *simulation-ddradseq.sh*: This script simulates reads of a ddRADseq obtained from previously generated fragments in the digest of the genomes of *Saccharomyces cerevisiae*, *Homo sapiens* and *Pinus taeda* modifying two options: the number of reads to be generated and the probability of loci bearing PCR duplicates.
- *simulation-pcrdupprob.sh*: This script analyses the effect of PCR duplicates probability on the number of reads to be generated.
- *simulation-gcfactor.sh*: This script analyses the effect of the GC factor on the number of reads to be generated and on the probability of loci bearing PCR duplicates.
- *simulation-pipeline.sh*: This script generates fragments from a genome, simulates a double digest, generates their reads, removes PCR duplicates, demultiplexes the individuals, trims the adapters and other Illumina specific sequences, aligns the reads and gets SAM, BAM, BED and VCF format files to study and visualize alignments. BWA is used to align, and SAMtools, BEDtools and VCFtools to get files.

The scripts are shown below. For clarity, each option is set when a ddRADseqTools program is called, although it is not necessary if their values are equal to the values stored in the configuration files (see chapter Common characteristics of programs).

These scripts have to be adapted to the conditions of the experiment that are been simulated, and suited to the directory address where the software package is uncompressed. E. g. in the *simulation-test.sh* there are options that are common and affect to several programs. Then, instead of repeating the value of each program call, a variable is used. They are assigned in “Set run environment” step and their values are marked in green. They are:

- The variables DDRADSEQTOOLSDIR, GENOMESDIR, FRAGSDIR, READSDIR and STATSDIR contain the directories where ddRADseqTools programs were uncompressed, genomes were downloaded, reads will be generated and the statistics will be written, respectively. In *simulation-test.sh* they are subdirectories of the directory ddRADseqTools in the home directory of the user.
- The variable GENOME contains the reference genome to study. In *simulation-test.sh* the genome assigned is the value of GENOME_SCEREVISIAE, i. e. GCF_000146045.2_R64_genomic.fna.gz. Its value must be modified to the proper genome file.

- The variables ENZYME1, ENZYME2, TECHNIQUE, FORMAT, READTYPE, INDEX1LEN, INDEX2LEN, DBRLEN, WEND and CEND contain the name of the first restriction enzyme or its restriction site sequence, the name of the second restriction enzyme or its restriction site sequence, ddRADseq methodology, the format of the output file, the read type, the index sequence length in the adapter 1, the index sequence length in the adapter 2, the DBR length, the end code where the adapter 1 is, and the end code where the adapter 2 is, respectively.

Other options are used in one program only, or they are static and usually do not vary from run to run. The values for these options are marked in yellow.

Two steps are needed to perform the trimming of the read ends contained in the two files of each individual (already demultiplexed) with *readstrim.py*: 1) to get a list of the read files containing the sequence of adaptor 1; and 2) to run a loop *while* the files exist in order to treat each listed file. In order to do this task, we use three variables:

a) FILE_1 is a variable that contains the name of these files.

b) FILE_2 is a second variable needed for PE reads (as in the example script). It contains the name of the read files that include adaptor 2.

c) FILE_TRIMMED contains the names of the output files.

The value for FILE_2 and FILE_TRIMMED are built using standard UNIX commands (*sed* and *echo)*. An example of this is marked in turquoise.

simulation-test-se-fastq.sh

```
#!/bin/bash

#-------------------------------------------------------------------------------

# This script executes a test of each program of the software package ddRADseqTools

#-------------------------------------------------------------------------------

# Control parameters

if [ -n "$*" ]; then echo 'This script has not parameters.'; exit 1; fi

#-------------------------------------------------------------------------------

# Set run environment

DDRADSEQTOOLSDIR=~/ddRADseqTools/Exe    # ddRADseqTools programs directory
GENOMESDIR=~/ddRADseqTools/Genomes      # genomes file directory
FRAGSDIR=~/ddRADseqTools/Fragments      # fragments directory
READSDIR=~/ddRADseqTools/Reads          # reads directory
STATSDIR=~/ddRADseqTools/Statistics     # statistics directory

if [ ! -d "$FRAGSDIR" ]; then mkdir $FRAGSDIR; fi
if [ ! -d "$READSDIR" ]; then mkdir $READSDIR; else rm -f $READSDIR/*; fi
if [ ! -d "$STATSDIR" ]; then mkdir $STATSDIR; else rm -f $STATSDIR/*; fi

GENOME_SCEREVISIAE=GCF_000146045.2_R64_genomic.fna.gz
GENOME_CELEGANS=GCF_000002985.6_WBcel235_genomic.fna.gz
GENOME_DMELANOGASTER=GCF_000001215.4_Release_6_plus_ISO1_MT_genomic.fna.gz
GENOME_HSAPIENS=GCF_000001405.29_GRCh38.p3_genomic.fna.gz
GENOME_QROBUR=ena.fasta
GENOME_PTAEDA=ptaeda.v1.01.scaffolds.fasta.gz

GENOME=$GENOME_SCEREVISIAE    # genome used in this run

ENZYME1=EcoRI
ENZYME2=MseI
TECHNIQUE=IND1_IND2_DBR
FORMAT=FASTQ
READTYPE=PE
INDEX1LEN=6
INDEX2LEN=6
DBRLEN=4
WEND=end31
CEND=end32
INDIVIDUALSFILE=individuals-8index1-6index2.txt

if [ `ulimit -n` -lt 1024 ]; then ulimit -n 1024; fi

#-------------------------------------------------------------------------------



# Generate genome fragments and get statistics

echo '**************************************************'
echo 'GENOME FRAGMENTS ARE BEING GENERATED FROM GENOME ...'

$DDRADSEQTOOLSDIR/rsitesearch.py \
   --genfile=$GENOMESDIR/$GENOME \
   --fragsfile=$FRAGSDIR/fragments-genome.fasta \
   --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
   --enzyme1=$ENZYME1 \
   --enzyme2=$ENZYME2 \
   --minfragsize=101 \
   --maxfragsize=300 \
   --fragstfile=$STATSDIR/fragments-genome-stats.txt \
   --fragstinterval=25
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#-------------------------------------------------------------------------------

# Generate random fragments and get statistics

echo '**************************************************'
echo 'GENOME FRAGMENTS ARE BEING GENERATED RANDOMLY ...'

$DDRADSEQTOOLSDIR/fragsgeneration.py \
   --fragsfile=$FRAGSDIR/fragments-random.fasta \
   --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
   --enzyme1=$ENZYME1 \
   --enzyme2=$ENZYME2 \
   --fragsnum=3103 \
   --minfragsize=101 \
   --maxfragsize=300 \
   --fragstfile=$STATSDIR/fragments-random-stats.txt \
   --fragstinterval=25
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#-------------------------------------------------------------------------------

# Locate several sequences into the genome

echo '**************************************************'
echo 'SEVERAL SEQUENCES ARE BEING LOCATED ...'

$DDRADSEQTOOLSDIR/seqlocation.py \
   --genfile=$GENOMESDIR/$GENOME \
   --seq=TGGGTGGAACTAGTAGCTGGAGATGCGTTCTAAAGGATCTAAAATCAGACTCACCCCAAAAACTGGGT
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

$DDRADSEQTOOLSDIR/seqlocation.py \
   --genfile=$GENOMESDIR/$GENOME \
   --seq=AAGAACATCTCGAAGCCAGAATTGAGCATCATATATTCGAGCTGTACAAACATCATGGCCTACAAGAA
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

$DDRADSEQTOOLSDIR/seqlocation.py \
   --genfile=$GENOMESDIR/$GENOME \
   --seq=GAAGTAGTGTACCACATTTGTAAGTTTAGATGCCTATTGGAAATGAGCGGGTACAAAAATGACGAAGT
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi


$DDRADSEQTOOLSDIR/seqlocation.py \
   --genfile=$GENOMESDIR/$GENOME \
   --seq=TTTATAATCCAGACCTCCCAAAAGAGGCAATCGTCAACTTCTGTCAATCTATTCTAGATGCTA
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#-------------------------------------------------------------------------------

# Generate ddRADseq simulated reads

echo '**************************************************'
echo 'DDRADSEQ SIMULATED READS ARE BEING GENERATED ...'

$DDRADSEQTOOLSDIR/simddradseq.py \
   --fragsfile=$FRAGSDIR/fragments-genome.fasta \
   --technique=$TECHNIQUE \
   --format=$FORMAT \
   --readsfile=$READSDIR/reads \
   --readtype=$READTYPE \
   --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
   --enzyme1=$ENZYME1 \
   --enzyme2=$ENZYME2 \
   --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
   --index1len=$INDEX1LEN \
   --index2len=$INDEX2LEN \
   --dbrlen=$DBRLEN \
   --wend=$WEND \
   --cend=$CEND \
   --individualsfile=$DDRADSEQTOOLSDIR/$INDIVIDUALSFILE \
   --locinum=3000 \
   --readsnum=300000 \
   --minreadvar=0.8 \
   --maxreadvar=1.2 \
   --insertlen=100 \
   --mutprob=0.2 \
   --locusmaxmut=1 \
   --indelprob=0.1 \
   --maxindelsize=10 \
   --dropout=0.0 \
   --pcrdupprob=0.2 \
   --pcrdistribution=MULTINOMIAL \
   --multiparam=0.167,0.152,0.136,0.121,0.106,0.091,0.076,0.061,0.045,0.030,0.015 \
   --poissonparam=1.0 \
   --gcfactor=0.2
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#-------------------------------------------------------------------------------

# Remove the PCR duplicates

echo '**************************************************'
echo 'THE PRC DUPLICATES ARE BEING REMOVED ...'

$DDRADSEQTOOLSDIR/pcrdupremoval.py \
   --format=$FORMAT \
   --readtype=$READTYPE \
   --readsfile1=$READSDIR/reads-1.fastq \
   --readsfile2=$READSDIR/reads-2.fastq \
   --clearfile=$READSDIR/reads-cleared \
   --dupstfile=$STATSDIR/pcrduplicates-stats.txt
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#-------------------------------------------------------------------------------

# Demultiplex the individual files

echo '**************************************************'
echo 'INDIVIDUAL FILES ARE BEING DEMULTIPLEXED ...'

$DDRADSEQTOOLSDIR/indsdemultiplexing.py \
   --technique=$TECHNIQUE \
   --format=$FORMAT \
   --readtype=$READTYPE \
   --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
   --index1len=$INDEX1LEN \
   --index2len=$INDEX2LEN \
   --dbrlen=$DBRLEN \
   --wend=$WEND \
   --cend=$CEND \
   --individualsfile=$DDRADSEQTOOLSDIR/$INDIVIDUALSFILE \
   --readsfile1=$READSDIR/reads-cleared-1.fastq \
   --readsfile2=$READSDIR/reads-cleared-2.fastq
```

```
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#-------------------------------------------------------------------------------


# Trim adapters and other Illumina-specific sequences from reads

echo '**************************************************'
echo 'ADAPTERS AND OTHER ILLUMINA SPECIFIC SEQUENCES ARE BEING TRIMMED ...'

ls $READSDIR/demultiplexed-ind*-1.fastq > $READSDIR/reads-files.txt

while read FILE_1; do

   FILE_2=`echo $FILE_1 | sed 's/-1.fastq/-2.fastq/g'`
   FILE_TRIMMED=`echo $FILE_1 | sed 's/-1.fastq/-trimmed/g'`

   $DDRADSEQTOOLSDIR/readstrim.py \
      --technique=$TECHNIQUE \
      --format=$FORMAT \
      --readtype=$READTYPE \
      --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
      --index1len=$INDEX1LEN \
      --index2len=$INDEX2LEN \
      --dbrlen=$DBRLEN \
      --wend=$WEND \
      --cend=$CEND \
      --readsfile1=$FILE_1 \
      --readsfile2=$FILE_2 \
      --trimfile=$FILE_TRIMMED
   if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

done < $READSDIR/reads-files.txt

#-------------------------------------------------------------------------------

# End
echo '**************************************************'
exit 0

#-------------------------------------------------------------------------------
```

simulation-genomes.sh

```
#!/bin/bash

#-------------------------------------------------------------------------------

# This script analyses how several enzymes pairs perform a double digest of the
# genome of Saccharomyces cerevisiae, Homo sapiens and Pinus taeda

#-------------------------------------------------------------------------------

# Control parameters

if [ -n "$*" ]; then echo 'This script has not parameters'; exit 1; fi

#-------------------------------------------------------------------------------

# Set run environment

DDRADSEQTOOLSDIR=~/ddRADseqTools/Exe    # ddRADseqTools programs directory
GENOMESDIR=~/ddRADseqTools/Genomes      # genomes file directory
FRAGSDIR=~/ddRADseqTools/Fragments      # fragments directory
STATSDIR=~/ddRADseqTools/Statistics     # statistics directory

if [ ! -d "$FRAGSDIR" ]; then mkdir $FRAGSDIR; else rm -f $FRAGSDIR/*; fi
if [ ! -d "$STATSDIR" ]; then mkdir $STATSDIR; else rm -f $STATSDIR/*; fi

SCEREVISIAE='Scerevisiae'
SCEREVISIAE_GENOME=GCF_000146045.2_R64_genomic.fna.gz

HSAPIENS='Hsapiens'
HSAPIENS_GENOME=GCF_000001405.29_GRCh38.p3_genomic.fna.gz

PTAEDA='Ptaeda'
PTAEDA_GENOME=ptaeda.v1.01.scaffolds.fasta.gz

ENZYME1=(EcoRI SbfI PstI)    # codes of 1st restriction enzyme to study
ENZYME2=(MseI)               # codes of 2nd restriction enzyme to study

#-------------------------------------------------------------------------------

# Generate Saccharomyces cerevisiae genome fragments

echo '**************************************************'
echo 'SACCHAROMYCES CEREVISIAE - GENOME FRAGMENTS ARE BEING GENERATED ...'

for I in "${ENZYME1[@]}"
do
   for J in "${ENZYME2[@]}"
   do

      echo '--------------------------------------------------'
      echo "SIMULATION WITH ENZYME1=$I AND ENZYME2=$J ..."

      $DDRADSEQTOOLSDIR/rsitesearch.py \
         --genfile=$GENOMESDIR/$SCEREVISIAE_GENOME \
         --fragsfile=$FRAGSDIR/$SCEREVISIAE'-fragments-'$I'-'$J'.fasta' \
         --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
         --enzyme1=$I \
         --enzyme2=$J \
         --minfragsize=101 \
         --maxfragsize=300 \
         --fragstfile=$STATSDIR/$SCEREVISIAE'-fragments-'$I'-'$J'-stats.txt'\
         --fragstinterval=25
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   done
done

#-------------------------------------------------------------------------------

# Generate Homo sapiens genome fragments

echo '**************************************************'
echo 'HOMO SAPIENS - FRAGMENTS ARE BEING GENERATED ...'

for I in "${ENZYME1[@]}"
do
   for J in "${ENZYME2[@]}"
   do

      echo '--------------------------------------------------'
      echo "SIMULATION WITH ENZYME1=$I AND ENZYME2=$J ..."

      $DDRADSEQTOOLSDIR/rsitesearch.py \
         --genfile=$GENOMESDIR/$HSAPIENS_GENOME \
         --fragsfile=$FRAGSDIR/$HSAPIENS'-fragments-'$I'-'$J'.fasta' \
         --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
         --enzyme1=$I \
         --enzyme2=$J \
         --minfragsize=201 \
         --maxfragsize=300 \
         --fragstfile=$STATSDIR/$HSAPIENS'-fragments-'$I'-'$J'-stats.txt' \
         --fragstinterval=25
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   done
done

#-------------------------------------------------------------------------------

# Generate Pinus taeda genome fragments

echo '**************************************************'
echo 'PINUS TAEDA - GENOME FRAGMENTS ARE BEING GENERATED ...'

for I in "${ENZYME1[@]}"
do
   for J in "${ENZYME2[@]}"
   do

      echo '--------------------------------------------------'
      echo "SIMULATION WITH ENZYME1=$I AND ENZYME2=$J ..."

      $DDRADSEQTOOLSDIR/rsitesearch.py \
         --genfile=$GENOMESDIR/$PTAEDA_GENOME \
         --fragsfile=$FRAGSDIR/$PTAEDA'-fragments-'$I'-'$J'.fasta' \
         --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
         --enzyme1=$I \
         --enzyme2=$J \
         --minfragsize=201 \
         --maxfragsize=300 \
         --fragstfile=$STATSDIR/$PTAEDA'-fragments-'$I'-'$J'-stats.txt' \
         --fragstinterval=25
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   done
done

#-------------------------------------------------------------------------------
```

```
# End
echo '**************************************************'
exit 0

#--------------------------------------------------------------------------------
```

simulation-ddradseq.sh

```
#!/bin/bash

#-------------------------------------------------------------------------------

# This script simulates reads of a ddRADseq gotten from fragments previously
# obtained of the digest of the genome of Saccharomyces cerevisiae, Homo
# sapiens and Pinus taeda varying two options: number of reads to generate
# and probability of loci bearing PCR duplicates

#-------------------------------------------------------------------------------

# Control parameters

if [ -n "$*" ]; then echo 'This script has not parameters'; exit 1; fi

#-------------------------------------------------------------------------------

# Set run environment

DDRADSEQTOOLS=~/ddRADseqTools/Exe        # ddRADseqTools programs directory
FRAGSDIR=~/ddRADseqTools/Fragments       # fragments directory
READSDIR=~/ddRADseqTools/Reads           # reads directory
STATSDIR=~/ddRADseqTools/Statistics      # statistics directory

if [ ! -d "$READSDIR" ]; then mkdir $READSDIR; else rm -f $READSDIR/*; fi
if [ ! -d "$STATSDIR" ]; then mkdir $STATSDIR; else rm -f $STATSDIR/*; fi

SCEREVISIAE='Scerevisiae'
SCEREVISIAE_FRAGS_FILE=$SCEREVISIAE'-fragments-EcoRI-MseI.fasta'
SCEREVISIAE_READSNUM=(300000 600000 1200000 2400000)
SCEREVISIAE_PCRDUPPROB=(0.2 0.4 0.6)

HSAPIENS='Hsapiens'
HSAPIENS_FRAGS_FILE=$HSAPIENS'-fragments-SbfI-MseI.fasta'
HSAPIENS_READSNUM=(2000000 4000000 8100000 16100000)
HSAPIENS_PCRDUPPROB=(0.2 0.4 0.6)

PTAEDA='Ptaeda'
PTAEDA_FRAGS_FILE=$PTAEDA'-fragments-SbfI-MseI.fasta'
PTAEDA_READSNUM=(2500000 5100000 10200000 20400000)
PTAEDA_PCRDUPPROB=(0.2 0.4 0.6)

TECHNIQUE=IND1_IND2_DBR
FORMAT=FASTQ
READTYPE=PE
INDEX1LEN=6
INDEX2LEN=6
DBRLEN=4
WEND=end31
CEND=end32
INDIVIDUALSFILE=individuals-8index1-6index2.txt

if [ `ulimit -n` -lt 1024 ]; then ulimit -n 1024; fi

#-------------------------------------------------------------------------------

# Saccharomyces cerevisiae

echo '**************************************************'
echo 'SACCHAROMYCES CEREVISIAE'
```

```
for I in "${SCEREVISIAE_READSNUM[@]}"
do
   for J in "${SCEREVISIAE_PCRDUPPROB[@]}"
   do

      # Generate ddRADseq simulated reads

      echo '---------------------------------------------------'
      echo "SIMULATED READS ARE BEING GENERATED WITH READSNUM=$I AND PCRDUPPROB=$J ..."

      $DDRADSEQTOOLS/simddradseq.py \
         --fragsfile=$FRAGSDIR/$SCEREVISIAE_FRAGS_FILE \
         --technique=$TECHNIQUE \
         --format=$FORMAT \
         --readsfile=$READSDIR/$SCEREVISIAE'-reads-'$I'-'$J \
         --readtype=$READTYPE \
         --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
         --enzyme1=EcoRI \
         --enzyme2=MseI \
         --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
         --index1len=$INDEX1LEN \
         --index2len=$INDEX2LEN \
         --dbrlen=$DBRLEN \
         --wend=$WEND \
         --cend=$CEND \
         --individualsfile=$DDRADSEQTOOLSDIR/$INDIVIDUALSFILE \
         --locinum=3000 \
         --readsnum=$I \
         --minreadvar=0.8 \
         --maxreadvar=1.2 \
         --insertlen=100 \
         --mutprob=0.2 \
         --locusmaxmut=1 \
         --indelprob=0.1 \
         --maxindelsize=10 \
         --dropout=0.0 \
         --pcrdupprob=$J \
         --pcrdistribution=MULTINOMIAL \
         --multiparam=0.167,0.152,0.136,0.121,0.106,0.091,0.076,0.061,0.045,0.030,0.015 \
         --poissonparam=1.0 \
         --gcfactor=0.2
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

      # Remove the PCR duplicates

      echo '---------------------------------------------------'
      echo "REMOVING PCR DUPLICATES ..."

      $DDRADSEQTOOLS/pcrdupremoval.py \
         --readsfile1=$READSDIR/$SCEREVISIAE'-reads-'$I'-'$J'-1.fastq' \
         --readsfile2=$READSDIR/$SCEREVISIAE'-reads-'$I'-'$J'-2.fastq' \
         --clearfile=$READSDIR/$SCEREVISIAE'-reads-cleared-'$I'-'$J \
         --dupstfile=$STATSDIR/$SCEREVISIAE'-pcrduplicates-stats-'$I'-'$J'.txt'
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   done
done

#--------------------------------------------------------------------------------

# Homo sapiens

echo '***************************************************'
echo 'HOMO SAPIENS'
```

```
for I in "${HSAPIENS_READSNUM[@]}"
do
   for J in "${HSAPIENS_PCRDUPPROB[@]}"
   do



   # Generate ddRADseq simulated reads

   echo '---------------------------------------------------'
   echo "SIMULATED READS ARE BEING GENERATED WITH READSNUM=$I AND PCRDUPPROB=$J ..."

      $DDRADSEQTOOLS/simddradseq.py \
         --fragsfile=$FRAGSDIR/$HSAPIENS_FRAGS_FILE \
         --technique=$TECHNIQUE \
         --format=$FORMAT \
         --readsfile=$READSDIR/$HSAPIENS'-reads-'$I'-'$J \
         --readtype=$READTYPE \
         --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
         --enzyme1=SbfI \
         --enzyme2=MseI \
         --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
         --index1len=$INDEX1LEN \
         --index2len=$INDEX2LEN \
         --dbrlen=$DBRLEN \
         --wend=$WEND \
         --cend=$CEND \
         --individualsfile=$DDRADSEQTOOLSDIR/$INDIVIDUALSFILE \
         --locinum=20700 \
         --readsnum=$I \
         --minreadvar=0.8 \
         --maxreadvar=1.2 \
         --insertlen=100 \
         --mutprob=0.2 \
         --locusmaxmut=1 \
         --indelprob=0.1 \
         --maxindelsize=10 \
         --dropout=0.0 \
         --pcrdupprob=$J \
         --pcrdistribution=MULTINOMIAL \
         --multiparam=0.167,0.152,0.136,0.121,0.106,0.091,0.076,0.061,0.045,0.030,0.015 \
         --poissonparam=1.0 \
         --gcfactor=0.2
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

      # Remove the PCR duplicates

      echo '---------------------------------------------------'
      echo "REMOVING PCR DUPLICATES ..."

      $DDRADSEQTOOLS/pcrdupremoval.py \
         --readsfile1=$READSDIR/$HSAPIENS'-reads-'$I'-'$J'-1.fastq' \
         --readsfile2=$READSDIR/$HSAPIENS'-reads-'$I'-'$J'-2.fastq' \
         --clearfile=$READSDIR/$HSAPIENS'-reads-cleared-'$I'-'$J \
         --dupstfile=$STATSDIR/$HSAPIENS'-pcrduplicates-stats-'$I'-'$J'.txt'
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   done
done

#-------------------------------------------------------------------------------

# Pinus taeda
```

```
echo '**************************************************'
echo 'PINUS TAEDA'

for I in "${PTAEDA_READSNUM[@]}"
do
   for J in "${PTAEDA_PCRDUPPROB[@]}"
   do

      # Generate ddRADseq simulated reads

      echo '--------------------------------------------------'
      echo "SIMULATED READS ARE BEING GENERATED WITH READSNUM=$I AND PCRDUPPROB=$J ..."

      $DDRADSEQTOOLS/simddradseq.py \
         --fragsfile=$FRAGSDIR/$PTAEDA_FRAGS_FILE \
         --technique=$TECHNIQUE \
         --format=$FORMAT \
         --readsfile=$READSDIR/$PTAEDA'-reads-'$I'-'$J \
         --readtype=$READTYPE \
         --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
         --enzyme1=SbfI \
         --enzyme2=MseI \
         --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
         --index1len=$INDEX1LEN \
         --index2len=$INDEX2LEN \
         --dbrlen=$DBRLEN \
         --wend=$WEND \
         --cend=$CEND \
         --individualsfile=$DDRADSEQTOOLSDIR/$INDIVIDUALSFILE \
         --locinum=26000 \
         --readsnum=$I \
         --minreadvar=0.8 \
         --maxreadvar=1.2 \
         --insertlen=100 \
         --mutprob=0.2 \
         --locusmaxmut=1 \
         --indelprob=0.1 \
         --maxindelsize=10 \
         --dropout=0.0 \
         --pcrdupprob=$J \
         --pcrdistribution=MULTINOMIAL \
         --multiparam=0.167,0.152,0.136,0.121,0.106,0.091,0.076,0.061,0.045,0.030,0.015 \
         --poissonparam=1.0 \
         --gcfactor=0.2
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi


      # Remove the PCR duplicates

      echo '--------------------------------------------------'
      echo "REMOVING PCR DUPLICATES ..."

      $DDRADSEQTOOLS/pcrdupremoval.py \
         --readsfile1=$READSDIR/$PTAEDA'-reads-'$I'-'$J'-1.fastq' \
         --readsfile2=$READSDIR/$PTAEDA'-reads-'$I'-'$J'-2.fastq' \
         --clearfile=$READSDIR/$PTAEDA'-reads-cleared-'$I'-'$J \
         --dupstfile=$STATSDIR/$PTAEDA'-pcrduplicates-stats-'$I'-'$J'.txt'
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   done
done

#-------------------------------------------------------------------------------
```

```
# End
echo '**************************************************'
exit 0

#-------------------------------------------------------------------------------
```

simulation-pcrdupprob.sh

```
#!/bin/bash

#-------------------------------------------------------------------------------

# This script analyses the effect of the probability of PCR duplicates on the number
# of reads to be generated

#-------------------------------------------------------------------------------

# Control parameters

if [ -n "$*" ]; then echo 'This script has not parameters'; exit 1; fi

#-------------------------------------------------------------------------------

# Set run environment

DDRADSEQTOOLS=$TRABAJO/ddRADseqTools/Exe      # ddRADseqTools programs directory
FRAGSDIR=$TRABAJO/ddRADseqTools/Fragments     # fragments directory
READSDIR=$TRABAJO/ddRADseqTools/Reads         # reads directory
STATSDIR=$TRABAJO/ddRADseqTools/Statistics    # statistics directory

if [ ! -d "$READSDIR" ]; then mkdir $READSDIR; else rm -f $READSDIR/*; fi
if [ ! -d "$STATSDIR" ]; then mkdir $STATSDIR; else rm -f $STATSDIR/*; fi

SCEREVISIAE='Scerevisiae'
SCEREVISIAE_FRAGS_FILE=$SCEREVISIAE'-fragments-EcoRI-MseI.fasta'
SCEREVISIAE_READSNUM=(300000 600000 1200000 2400000)
SCEREVISIAE_PCRDUPPROB=(0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9)

TECHNIQUE=IND1_IND2_DBR
FORMAT=FASTQ
READTYPE=PE
INDEX1LEN=6
INDEX2LEN=6
DBRLEN=4
WEND=end31
CEND=end32
INDIVIDUALSFILE=individuals-8index1-6index2.txt

if [ `ulimit -n` -lt 1024 ]; then ulimit -n 1024; fi

#-------------------------------------------------------------------------------

# Saccharomyces cerevisiae

echo '**************************************************'
echo 'SACCHAROMYCES CEREVISIAE'

for I in "${SCEREVISIAE_READSNUM[@]}"
do
   for J in "${SCEREVISIAE_PCRDUPPROB[@]}"
   do

      # Generate ddRADseq simulated reads

      echo '--------------------------------------------------'
      echo "SIMULATED READS ARE BEING GENERATED WITH READSNUM=$I AND PCRDUPPROB=$J ..."
      $DDRADSEQTOOLS/simddradseq.py \
         --fragsfile=$FRAGSDIR/$SCEREVISIAE_FRAGS_FILE \
         --technique=$TECHNIQUE \
         --format=$FORMAT \
         --readsfile=$READSDIR/$SCEREVISIAE'-reads-'$I'-'$J \
         --readtype=$READTYPE \
         --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
         --enzyme1=EcoRI \
         --enzyme2=MseI \
         --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
         --index1len=$INDEX1LEN \
         --index2len=$INDEX2LEN \
         --dbrlen=$DBRLEN \
         --wend=$WEND \
         --cend=$CEND \
         --individualsfile=$DDRADSEQTOOLSDIR/$INDIVIDUALSFILE \
         --locinum=3000 \
         --readsnum=$I \
         --minreadvar=0.8 \
         --maxreadvar=1.2 \
         --insertlen=100 \
         --mutprob=0.2 \
         --locusmaxmut=1 \
         --indelprob=0.1 \
         --maxindelsize=10 \
         --dropout=0.0 \
         --pcrdupprob=$J \
         --pcrdistribution=MULTINOMIAL \
         --multiparam=0.167,0.152,0.136,0.121,0.106,0.091,0.076,0.061,0.045,0.030,0.015 \
         --poissonparam=1.0 \
         --gcfactor=0.2
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

      # Remove the PCR duplicates

      echo '--------------------------------------------------'
      echo "REMOVING PCR DUPLICATES ..."

      $DDRADSEQTOOLS/pcrdupremoval.py \
         --format=$FORMAT \
         --readtype=$READTYPE \
         --readsfile1=$READSDIR/$SCEREVISIAE'-reads-'$I'-'$J'-1.fastq' \
         --readsfile2=$READSDIR/$SCEREVISIAE'-reads-'$I'-'$J'-2.fastq' \
         --clearfile=$READSDIR/$SCEREVISIAE'-reads-cleared-'$I'-'$J \
         --dupstfile=$STATSDIR/$SCEREVISIAE'-pcrduplicates-stats-'$I'-'$J'.txt'
      if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   done
done

#-------------------------------------------------------------------------------

# End
echo '**************************************************'
exit 0

#-------------------------------------------------------------------------------
```

simulation-gcfactor.sh

```
#!/bin/bash

#-------------------------------------------------------------------------------

# This script analyses the effect of the GC factor on the number of reads to
# be generated and the probability of loci bearing PCR duplicates

#-------------------------------------------------------------------------------

# Control parameters

if [ -n "$*" ]; then echo 'This script has not parameters'; exit 1; fi

#-------------------------------------------------------------------------------

# Set run environment

DDRADSEQTOOLS=~/ddRADseqTools/Exe      # ddRADseqTools programs directory
FRAGSDIR=~/ddRADseqTools/Fragments     # fragments directory
READSDIR=~/ddRADseqTools/Reads         # reads directory
STATSDIR=~/ddRADseqTools/Statistics    # statistics directory

if [ ! -d "$READSDIR" ]; then mkdir $READSDIR; else rm -f $READSDIR/*; fi
if [ ! -d "$STATSDIR" ]; then mkdir $STATSDIR; else rm -f $STATSDIR/*; fi

SCEREVISIAE='Scerevisiae'
SCEREVISIAE_FRAGS_FILE=$SCEREVISIAE'-fragments-EcoRI-MseI.fasta'
SCEREVISIAE_READSNUM=(600000 1200000)
SCEREVISIAE_PCRDUPPROB=(0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9)
SCEREVISIAE_GCFACTOR=(0.0 0.1 0.2 0.3 0.4 0.5)

TECHNIQUE=IND1_IND2_DBR
FORMAT=FASTQ
READTYPE=PE
INDEX1LEN=6
INDEX2LEN=6
DBRLEN=4
WEND=end31
CEND=end32
INDIVIDUALSFILE=individuals-8index1-6index2.txt

if [ `ulimit -n` -lt 1024 ]; then ulimit -n 1024; fi

#-------------------------------------------------------------------------------

# Generate ddRADseq simulated reads of Saccharomyces cerevisiae

echo '**************************************************'
echo 'SACCHAROMYCES CEREVISIAE'

for I in "${SCEREVISIAE_READSNUM[@]}"
do
   for J in "${SCEREVISIAE_PCRDUPPROB[@]}"
   do
      for K in "${SCEREVISIAE_GCFACTOR[@]}"
      do
```

```
         # Generate ddRADseq simulated reads

         echo '--------------------------------------------------'
         echo "SIMULATION WITH READSNUM=$I AND PCRDUPPROB=$J AND GCFACTOR=$K ..."

         $DDRADSEQTOOLS/simddradseq.py \
            --fragsfile=$FRAGSDIR/$SCEREVISIAE_FRAGS_FILE \
            --technique=$TECHNIQUE \
            --format=$FORMAT \
            --readsfile=$READSDIR/$SCEREVISIAE'-reads-'$I'-'$J'-'$K \
            --readtype=$READTYPE \
            --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
            --enzyme1=EcoRI \
            --enzyme2=MseI \
            --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
            --index1len=$INDEX1LEN \
            --index2len=$INDEX2LEN \
            --dbrlen=$DBRLEN \
            --wend=$WEND \
            --cend=$CEND \
            --individualsfile=$DDRADSEQTOOLSDIR/$INDIVIDUALSFILE \
            --locinum=3000 \
            --readsnum=$I \
            --minreadvar=0.8 \
            --maxreadvar=1.2 \
            --insertlen=100 \
            --mutprob=0.2 \
            --locusmaxmut=1 \
            --indelprob=0.1 \
            --maxindelsize=10 \
            --dropout=0.0 \
            --pcrdupprob=$J \
            --pcrdistribution=MULTINOMIAL \
            --multiparam=0.167,0.152,0.136,0.121,0.106,0.091,0.076,0.061,0.045,0.030,0.015 \
            --poissonparam=1.0 \
            --gcfactor=$K
         if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

         # Remove the PCR duplicates

         echo '--------------------------------------------------'
         echo "REMOVING PCR DUPLICATES ..."

         $DDRADSEQTOOLS/pcrdupremoval.py \
            --format=$FORMAT \
            --readtype=$READTYPE \
            --readsfile1=$READSDIR/$SCEREVISIAE'-reads-'$I'-'$J'-'$K'-1.fastq' \
            --readsfile2=$READSDIR/$SCEREVISIAE'-reads-'$I'-'$J'-'$K'-2.fastq' \
            --clearfile=$READSDIR/$SCEREVISIAE'-reads-cleared-'$I'-'$J'-'$K \
            --dupstfile=$STATSDIR/$SCEREVISIAE'-pcrduplicates-stats-'$I'-'$J'-'$K'.txt'
         if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

      done
   done
done

#-------------------------------------------------------------------------------

# End
echo '**************************************************'
exit 0

#-------------------------------------------------------------------------------
```

simulation-pipeline.sh

```
#!/bin/bash

#-------------------------------------------------------------------------------

# This script generates fragments from a genome, simulates a double digest,
# generates their reads, removes PCR duplicates, demultiplexes the individuals, trims
# the adapters and other Illumina specific sequences, aligns the reads and gets
# SAM, BAM, BED and VCF format files to study and visualize alignments

#-------------------------------------------------------------------------------

# WARNING

# This script uses the following bioinformatics tools:
# - BWA v0.7.12-r1039 (http://bio-bwa.sourceforge.net/)
# - SAMtools & BCFtools v0.1.19-96bsf2294a (http://www.htslib.org/)
# - BEDtools v2.17.0 (http://bedtools.readthedocs.io/)
# - VCFtools v0.1.11 (https://vcftools.github.io/)

#-------------------------------------------------------------------------------

# Control parameters

if [ -n "$*" ]; then echo 'This script has not parameters.'; exit 1; fi

#-------------------------------------------------------------------------------

# Set run environment

SAMTOOLSDIR=/usr/share/samtools    # SAMTools directory

DDRADSEQTOOLSDIR=~/ddRADseqTools/Exe    # ddRADseqTools programs directory
GENOMESDIR=~/ddRADseqTools/Genomes      # genomes file directory
FRAGSDIR=~/ddRADseqTools/Fragments      # fragments directory
READSDIR=~/ddRADseqTools/Reads          # reads directory
STATSDIR=~/ddRADseqTools/Statistics     # statistics directory
ALIGNDIR=~/ddRADseqTools/Alignments     # alignments directory

if [ ! -d "$FRAGSDIR" ]; then mkdir $FRAGSDIR; fi
if [ ! -d "$READSDIR" ]; then mkdir $READSDIR; else rm -f $READSDIR/*; fi
if [ ! -d "$STATSDIR" ]; then mkdir $STATSDIR; else rm -f $STATSDIR/*; fi
if [ ! -d "$ALIGNDIR" ]; then mkdir $ALIGNDIR; else rm -f $ALIGNDIR/*; fi

GENOME_SCEREVISIAE=GCF_000146045.2_R64_genomic.fna.gz
GENOME_CELEGANS=GCF_000002985.6_WBcel235_genomic.fna.gz
GENOME_DMELANOGASTERe=GCF_000001215.4_Release_6_plus_ISO1_MT_genomic.fna.gz
GENOME_HSAPIENS=GCF_000001405.29_GRCh38.p3_genomic.fna.gz
GENOME_QROBUR=ena.fasta
GENOME_PTAEDA=ptaeda.v1.01.scaffolds.fasta.gz

GENOME=$GENOME_SCEREVISIAE    # genome used in this run

ENZYME1=EcoRI
ENZYME2=MseI
TECHNIQUE=IND1_IND2_DBR
FORMAT=FASTQ
READTYPE=PE
INDEX1LEN=6
INDEX2LEN=6
DBRLEN=4
WEND=end31
CEND=end32
```

```
INDIVIDUALSFILE=individuals-8index1-6index2.txt

if [ `ulimit -n` -lt 1024 ]; then ulimit -n 1024; fi

#-------------------------------------------------------------------------------

# Generate genome fragments and get statistics

echo '**************************************************'
echo 'GENOME FRAGMENTS ARE BEING GENERATED FROM GENOME ...'

$DDRADSEQTOOLSDIR/rsitesearch.py \
   --genfile=$GENOMESDIR/$GENOME \
   --fragsfile=$FRAGSDIR/fragments-genome.fasta \
   --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
   --enzyme1=$ENZYME1 \
   --enzyme2=$ENZYME2 \
   --minfragsize=101 \
   --maxfragsize=300 \
   --fragstfile=$STATSDIR/fragments-genome-stats.txt \
   --fragstinterval=25
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#-------------------------------------------------------------------------------

# Generate ddRADseq simulated reads

echo '**************************************************'
echo 'DDRADSEQ SIMULATED READS ARE BEING GENERATED ...'

$DDRADSEQTOOLSDIR/simddradseq.py \
   --fragsfile=$FRAGSDIR/fragments-genome.fasta \
   --technique=$TECHNIQUE \
   --format=$FORMAT \
   --readsfile=$READSDIR/reads \
   --readtype=$READTYPE \
   --rsfile=$DDRADSEQTOOLSDIR/restrictionsites.txt \
   --enzyme1=$ENZYME1 \
   --enzyme2=$ENZYME2 \
   --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
   --index1len=$INDEX1LEN \
   --index2len=$INDEX2LEN \
   --dbrlen=$DBRLEN \
   --wend=$WEND \
   --cend=$CEND \
   --individualsfile=$DDRADSEQTOOLSDIR/$INDIVIDUALSFILE \
   --locinum=3000 \
   --readsnum=300000 \
   --minreadvar=0.8 \
   --maxreadvar=1.2 \
   --insertlen=100 \
   --mutprob=0.2 \
   --locusmaxmut=1 \
   --indelprob=0.1 \
   --maxindelsize=10 \
   --dropout=0.0 \
   --pcrdupprob=0.2 \
   --pcrdistribution=MULTINOMIAL \
   --multiparam=0.167,0.152,0.136,0.121,0.106,0.091,0.076,0.061,0.045,0.030,0.015 \
   --poissonparam=1.0 \
   --gcfactor=0.2
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#-------------------------------------------------------------------------------

# Remove the PCR duplicates
```

```
echo '**************************************************'
echo 'THE PRC DUPLICATES ARE BEING REMOVED ...'

$DDRADSEQTOOLSDIR/pcrdupremoval.py \
   --format=$FORMAT \
   --readtype=$READTYPE \
   --readsfile1=$READSDIR/reads-1.fastq \
   --readsfile2=$READSDIR/reads-2.fastq \
   --clearfile=$READSDIR/reads-cleared \
   --dupstfile=$STATSDIR/pcrduplicates-stats.txt
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#--------------------------------------------------------------------------------

# Demultiplex the individual files

echo '**************************************************'
echo 'INDIVIDUAL FILES ARE BEING DEMULTIPLEXED ...'

$DDRADSEQTOOLSDIR/indsdemultiplexing.py \
   --technique=$TECHNIQUE \
   --format=$FORMAT \
   --readtype=$READTYPE \
   --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
   --index1len=$INDEX1LEN \
   --index2len=$INDEX2LEN \
   --dbrlen=$DBRLEN \
   --wend=$WEND \
   --cend=$CEND \
   --individualsfile=$DDRADSEQTOOLSDIR/$INDIVIDUALSFILE \
   --readsfile1=$READSDIR/reads-cleared-1.fastq \
   --readsfile2=$READSDIR/reads-cleared-2.fastq
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#--------------------------------------------------------------------------------

# Trim adapters and other Illumina-specific sequences from reads

echo '**************************************************'
echo 'ADAPTERS AND OTHER ILLUMINA SPECIFIC SEQUENCES ARE BEING TRIMMED ...'

ls $READSDIR/demultiplexed-ind*-1.fastq > $READSDIR/reads-files.txt

while read FILE_1; do

   FILE_2=`echo $FILE_1 | sed 's/-1.fastq/-2.fastq/g'`
   FILE_TRIMMED=`echo $FILE_1 | sed 's/-1.fastq/-trimmed/g'`

   $DDRADSEQTOOLSDIR/readstrim.py \
      --technique=$TECHNIQUE \
      --format=$FORMAT \
      --readtype=$READTYPE \
      --endsfile=$DDRADSEQTOOLSDIR/ends.txt \
      --index1len=$INDEX1LEN \
      --index2len=$INDEX2LEN \
      --dbrlen=$DBRLEN \
      --wend=$WEND \
      --cend=$CEND \
      --readsfile1=$FILE_1 \
      --readsfile2=$FILE_2 \
      --trimfile=$FILE_TRIMMED
   if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

done < $READSDIR/reads-files.txt
```

```
#-------------------------------------------------------------------------------

# Index the genome

echo '**************************************************'
echo 'GENOME IS BEING INDEXED ...'

bwa index -a bwtsw $GENOMESDIR/$GENOME
if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

#-------------------------------------------------------------------------------

# Align sequences of individual files in SAM format

echo '**************************************************'
echo 'SEQUENCES OF INDIVIDUAL FILES ARE BEING ALIGNED IN SAM FORMAT ...'

ls $READSDIR/demultiplexed-ind*-trimmed-1.fastq > $READSDIR/reads-trimmed-files.txt

while read FILE1_TRIM; do

   FILE2_TRIM=`echo $FILE1_TRIM | sed 's/-trimmed-1.fastq/-trimmed-2.fastq/g'`
   FILE_SAM=`echo $FILE1_TRIM | sed 's/-1.fastq/.sam/g' | sed
"s|$READSDIR|$ALIGNDIR|g"`
   bwa mem $GENOMESDIR/$GENOME $FILE1_TRIM $FILE2_TRIM > $FILE_SAM
   if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

done < $READSDIR/reads-trimmed-files.txt

#-------------------------------------------------------------------------------

# Convert SAM files to BED, BAM and VCF format

echo '**************************************************'
echo 'SAM FILES ARE BEING CONVERTED IN BAM, BED AND VCF FORMAT ...'

ls $ALIGNDIR/*.sam > $ALIGNDIR/sam-files.txt

while read FILE_SAM; do

   FILE_BAM=`echo $FILE_SAM | sed 's/.sam/.bam/g'`
   FILE_BAM_STATS=`echo $FILE_SAM | sed 's/.sam/-stats-bam.txt/g'`
   FILE_BED=`echo $FILE_SAM | sed 's/.sam/.bed/g'`
   FILE_SORTED=`echo $FILE_SAM | sed 's/.sam/.sorted/g'`
   FILE_SORTED_BAM=`echo $FILE_SAM | sed 's/.sam/.sorted.bam/g'`
   FILE_VCF=`echo $FILE_SAM | sed 's/.sam/.vcf/g'`
   FILE_VCF_STATS1=`echo $FILE_SAM | sed 's/.sam/-stats-vcf-1.txt/g'`
   FILE_VCF_STATS2=`echo $FILE_SAM | sed 's/.sam/-stats-vcf-2.txt/g'`

   # convert SAM file to BAM format
   samtools import $GENOMESDIR/$GENOME.bwt $FILE_SAM $FILE_BAM
   if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   # get aligment statistic in a file
   samtools flagstat $FILE_BAM >$FILE_BAM_STATS
   if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   # convert BAM file to BED format
   bedtools bamtobed -i $FILE_BAM > $FILE_BED
   if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

   # sort the BAM file
   samtools sort $FILE_BAM $FILE_SORTED
   if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi
```

```
    # index the BAM file
    samtools index $FILE_SORTED_BAM
    if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

    # convert BAM file to VCF format
    samtools mpileup -uf  $GENOMESDIR/$GENOME $FILE_SORTED_BAM | bcftools view -vcg - > $FILE_VCF
    if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

    # get variant statistic in a file with SAMtools
    $SAMTOOLSDIR/vcfutils.pl qstats $FILE_VCF > $FILE_VCF_STATS1
    if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

    # get variant statistic in a file with VCFtools
    vcftools --vcf $FILE_VCF > $FILE_VCF_STATS2
    if [ $? -ne 0 ]; then echo 'Script ended with errors.'; exit 1; fi

done < $ALIGNDIR/sam-files.txt

#-------------------------------------------------------------------------------

# End
echo '**************************************************'
exit 0

#-------------------------------------------------------------------------------
```

References

Casbon JA, Osborne RJ, Brenner S, Lichtenstein CP (2011) A method for counting PCR template molecules with application to next-generation sequencing. *Nucleic Acids Research*, **39**, e81.

Mastretta-Yanes A, Arrigo N, Alvarez N, *et al*. (2015) Restriction site-associated DNA sequencing, genotyping error estimation and de novo assembly optimization for population genetic inference. *Molecular Ecology. Resources*, **15**, 28-41.

Peterson BK, Weber JN, Kay EH, Fisher HS, Hoekstra HE (2012) Double Digest RADseq: An Inexpensive Method for *De Novo* SNP Discovery and Genotyping in Model and Non-Model Species. *PLoS ONE*, **7**, e37135.

Schweyen H, Rozenberg A, Leese F (2014) Detection and Removal of PCR Duplicates in Population Genomic ddRAD Studies by Addition of a Degenerate Base Region (DBR) in Sequencing Adapters". *The Biological Bulletin*, **227**, 146-160.

Tin MMY, Rheindt FE, Cros E, Mikheyev AS (2015) Degenerate adaptor sequences for detecting PCR duplicates in reduced representation sequencing data improve genotype calling accuracy. *Molecular Ecology. Resources*, **15**, 329-336.