Title: ddRADseqTools: a software package for *in silico* simulation and testing of double digest RADseq experiments

Authors: F. Mora-Márquez[1], V. García-Olivares[2], B.C. Emerson[2,3], U. López de Heredia[1]

[1]Forest Genetics and Physiology Research Group, Technical University of Madrid (UPM), Ciudad Universitaria s/n, Madrid, Spain

[2]Island Ecology and Evolution Research Group, IPNA-CSIC, Tenerife, Canary Islands, Spain

[3]School of Biological Sciences, University of East Anglia, Norwich Research Park, Norwich NR4 7TJ, UK.

Keywords: allele dropout, coverage, ddRADseq, *in silico* simulation, PCR duplicates

Corresponding author: U. López de Heredia

Address: Forest Genetics and Physiology Research Group, Technical University of Madrid (UPM), Ciudad Universitaria s/n, Madrid, Spain.

Fax: +34 91 336 5556

Running title: *In silico* simulation of ddRADseq data

Abstract

Double digested RADseq (ddRADseq) is a NGS methodology that generates reads from thousands of loci targeted by restriction enzyme cut sites, across multiple individuals. To be statistically sound and economically optimal, a ddRADseq experiment has a preliminary design stage that needs to consider issues related to the selection of enzymes, particular features of the genome of the focal species, possible modifications to the library construction protocol, coverage needed to minimise missing data, and the potential sources of error that may impact upon the coverage. We present ddRADseqTools, a software package to help ddRADseq experimental design by (i) the generation of *in silico* double digested fragments, (ii) the construction of modified ddRADseq libraries using adapters with either one or two indexes and degenerate base regions (DBRs) to quantify PCR duplicates, and (iii) the initial steps of the bioinformatics pre-processing of reads. ddRADseqTools generates single-end (SE) or paired-end (PE) reads that may bear SNPs and/or indels. The effect of allele dropout and PCR duplicates on coverage is also simulated. The resulting output files can be submitted to pipelines of alignment and variant calling, in order to allow the fine-tuning of parameters. The software was validated with specific tests for the correct operability of the program. The correspondence between *in silico* settings and parameters from ddRADseq *in vitro* experiments was assessed to provide guidelines for the reliable performance of the software. ddRADseqTools is cost-efficient in terms of execution time, and can be run on computers with standard CPU and RAM configuration.

Introduction

Restriction site associated DNA sequencing (RADseq) is a fractional genome sequencing technology that allows for the cost effective genotyping of high numbers of individuals for a large number of polymorphisms (Baird *et al*. 2008; Davey & Blaxter 2010; Etter *et al*. 2011; Davey *et al*. 2011; Davey *et al*. 2013; Mastretta-Yanes *et al*. 2014). It has become popular in recent years because of its extraordinary potential for genetic mapping and population genetic studies in non-model species for which a reference genome is not available. Double digest restriction site associated DNA (ddRAD) sequencing, or ddRADseq, is a modification of RADseq that uses two restriction enzymes (Peterson *et al*. 2012), instead of only one. To obtain a manageable number of fragments, one enzyme typically has a rare motif while the other is more common, with the enzyme combination depending upon the size and structure of the target organism genome. The fragments produced by the ddRADseq platform are flanked by a cut site for each enzyme, and frequently fragments of a specific size range are selected to be sequenced. The fragments sequenced by ddRADseq consist of a genome insert between both restriction sites, and two ends that include an adapter and a primer. A short index sequence is attached to one or both ends to identify individuals. If a dual indexing approach is used (i.e. index sequences are embedded in both adaptors), the potential number of individuals that can be simultaneously sequenced increases considerably.

In vitro ddRADseq experiments may be optimized with preliminary *in silico* simulations. To achieve this, an effective *in silico* simulation tool must be able to generate plausible scenarios that take into account the different technical and analytical limitations that may compromise the success of an experiment. *In silico* ddRADseq approaches enable testing multiple scenarios to help in the design of the adapters, selection of optimal enzyme pair combinations, or the assessment of sufficient coverage to obtain sound results for the focal species, considering the biases produced by potential sources of error (see Mastretta-Yanes *et al*. 2015 for a review of the major sources of error in ddRADseq experiments). However, there are few available software tools that enable comprehensive *in silico* simulations for ddRADseq. The R package simRAD (Lepais & Weir 2014) provides functions to

simulate digestion and fragment selection, whereby a reference genome or randomly generated DNA sequences can be used as input for the digestion process. BU-RAD-seq (DaCosta & Sorenson 2014) is a RADseq data analysis pipeline that includes a program (Digital_RADs.py) for the digestion of a reference genome with one or two enzymes. Digital_RADs.py requires the motifs and the length of the down/upstream sequence (one enzyme) or the lower or upper size of the fragment (two enzymes). The Python program simRRLs included in the PyRAD pipeline (Eaton 2014) can be used to simulate RADseq-like random sequence data on a fixed species tree topology under a coalescent model. Although simRRLs is able to include some potential sources of error in the simulations, such as allele dropout or low coverage, it was not designed to handle reference genomes, and does not control for the presence of PCR duplicates.

Here, we describe ddRADseqTools, a software package for the design of ddRADseq experiments through the generation of *in silico* double digested single-end (SE) or paired-end (PE) read files under hypothetical scenarios of varying coverage and mutation rates. In addition to the selection of an optimal combination of enzymes and fragment size range for sequencing, the software takes into consideration two of the main potential sources of error present in ddRADSeq experiments that have a strong influence on coverage reduction - PCR duplicates and allele dropout- and parameterizes both for the simulation of ddRADseq read files. The output of the program includes the estimation of missing data produced by insufficient coverage, by both locus and individual. As such, experimental design can be optimized ~~in advance~~ to reduce bias in subsequent bioinformatic stages by running ddRADseqTools under different scenarios. The software is able to simulate modified ddRADseq libraries using adapters with either one or two indexes and degenerate base regions (DBRs) in one of the adapter ends to quantify PCR duplicates (Schweyen *et al.* 2014; Tin *et al.* 2015). The simulation of technical replicates to improve the accuracy of ddRADseq experiments (Mastretta-Yanes *et al*. 2015) is also possible. Technical replicates can detect and identify sources of variation in measurements, and limit the effect of spurious variation on hypothesis testing and parameter estimation (Blainey *et al*. 2014). Finally, ddRADseqTools also performs the initial steps of

bioinformatic pre-processing of ddRADseq reads: quantification and removal of PCR duplicates, demultiplexing of individuals, and trimming of adapters from raw reads. The resulting output files can be submitted to pipelines of alignment and variant calling for subsequent fine-tuning of parameters, to optimize and reduce ddRADseq experimental costs.

Methods

ddRADseqTools is a set of programs, configuration files and data for the design and *in silico* testing of ddRADseq experiments. ddRADseqTools is programmed in Python 3 (version 3.4 or higher is required), and runs on any computer with an Operative System (OS) that allows for Python 3: Linux/Unix, Mac OS X, Microsoft Windows and other OSs. The only dependencies required to run this software package are the *NumPy* (http://www.numpy.org/) and *matplotlib* (http://matplotlib.org/) libraries. The software package, along with its manual, is available from the software repository GitHub (https://github.com/GGFHF/ddRADseqTools).

Conceptual approaches

A flow-chart of the programs included in ddRADseqTools is shown in Figure 1. The work-flow has the three usual steps in an NGS experiment (Table 1): (1) library construction / *in silico* fragment generation; (2) high throughput sequencing / generation of simulated reads; (3) bioinformatic pre-processing of reads. The rationale behind the processes included in the code of ddRADSeqTools is discussed in the following sections.

Library construction / in silico fragments generation

A file of fragments is generated from a reference genome by *rsitesearch.py*; or fragment sequences are simulated randomly with *fragsgeneration.py*. If the genome-guided version of the software is used

(*rsitesearch.py*), a particular pair of restriction enzymes has to be specified and their action within the genome is simulated. Each fragment corresponds to a locus, and loci of a given size range can be selected to generate the read files. Size selection is a common strategy in ddRADseq experiments that allows stable shared region recovery across samples, and some control over the target number of loci for sequencing, thus facilitating coverage optimisation (Peterson *et al.* 2012).

High throughput sequencing / generation of simulated reads

Raw reads are generated by *simddradseq.py*. This program incorporates parameters for the type of library, number of reads, size of the genomic inserts, allele dropout probability, probability of loci bearing PCR duplicates, and mutation probability, that are set by the user. The software can simulate read files from any NGS platform, for either single-end (SE) or paired-end (PE) read files.

The ends of raw reads can be configured with flexibility, depending on the details of the type of ddRADseq library. The user may define specific adapters, *ad hoc* PCR primers, indexes at both ends of the read, and degenerate base regions (DBRs) according to the needs of the experiment and the sequencing platform of choice. As several modifications of the ddRADseq library construction methodology exist (e.g. Peterson *et al.* 2012; Mastretta-Yanes *et al.* 2015; Schweyen *et al*. 2014; Tin *et al*. 2015), this version of ddRADseqTools implements four of these techniques (Figure 2). In the original ddRADseq protocol (Peterson *et al.* 2012) a single index is used in *Adapter 1* to identify the individuals (Figure 2a). The number of samples that can be analysed in a single ddRADseq experiment can be increased by attaching two indexes to identify individuals (Figure 2b). The sequence of the end corresponding to *Adapter 1* includes an *index1* sequence, and the sequence of the end corresponding to *Adapter 2* includes an *index2* sequence. ddRADseqTools also considers design modifications of these two types of adapters by attaching a single index and a DBR to quantify PCR duplicates in *Adapter 1* (Figure 2c) (Schweyen *et al.* 2014; Tin *et al.* 2015); or using two indexes to identify individuals together with a DBR to quantify PCR duplicates (Figure 2d). The indexes and DBRs can have any size and be located at any position within the adapters.

Coverage is controlled by setting the number of loci, the number of individuals, and the total number of reads of the library. The average number of reads per locus is calculated by dividing the total number of reads to be generated, by the number of loci to sampled. Empirical data reported in the literature for diverse organisms show that coverage is unequal among loci and individuals. For instance, Recknagel *et al.* (2013) obtained an average coverage by locus and individual of 15x, with a standard deviation of 5.1x for fishes of genus *Amphilophus*. Mastretta-Yanes *et al.* (2014) reported an average coverage of 10.3x and a standard deviation of 4.2x for shrubs within the genus *Berberis*. With ddRADseqTools, unequal coverage is simulated by sampling the number of read copies at random for each locus and individual from a discrete uniform distribution. The minimum and maximum values of the distribution are defined by weighting the average number of reads per locus with two user defined parameters, *minreadvar* and *maxreadvar,* respectively, that vary between 0 and 1. If uniform coverage is desired, both options should be set to 1.

Loci affected by allele dropout are expected to show a lower coverage in ddRADseq experiments. Allele dropout may result in either no sequence data for an individual at a given locus, or for a heterozygote to be scored as a homozygote (Gautier *et al.* 2013), and affected alleles result in no reads. Allele dropout in ddRADseq may be produced by mutations at the enzyme recognition motif (Gautier *et al.* 2013), by DNA methylation in the case of methylation sensitive enzymes (Roberts et al. 2010), or by unequal PCR success (Casbon *et al.* 2011). In ddRADseqTools, the associated reduction in coverage is implemented as the probability of a locus to be affected by allele dropout. Under this approach, the higher the allele dropout probability, the higher the reduction in coverage and the generation of missing data. This parameter is independent of the probability of mutation in order to adjust to the variety of scenarios causing allele dropout.

PCR duplicates are artifacts of sequencing that derive from the attachment of more than one copy of the same original DNA molecule to different beads or cells. In ddRADseq experiments, these artifacts may inflate coverage estimates, or produce heterogeneous coverage distributions due to GC content and PCR bias. In ddRADseqTools, loci yielding PCR duplicates are selected at random

according to a probability defined by the user, which is modified by the GC ratio for each locus. Digested fragments with a higher GC ratio have a higher probability of producing PCR duplicates than those with a lower GC ratio (Davey *et al.* 2013). The number of duplicates per read is sampled from either a Poisson distribution, where the probability is controlled by the user with the parameter lambda; or by a multinomial distribution, for which a vector of probabilities for the number of duplicates by loci and individual must be introduced by the user.

Polymorphisms due to mutations (substitutions and/or indels) are incorporated within the simulated read files considering that individuals have two ~~fragment~~ sequences per locus (+ and - strands). Polymorphic states (one *mutated* and one *non-mutated*) are randomly assigned to + and - strands, conditioned upon a probability defined by the user that will be proportional to the average mutation rate for the organism, and that should not exceed 0.2. The number and type of mutations across the simulated reads are determined according to user-defined probabilities, as well as a maximum number of mutated positions per fragment. The nucleotide positions of mutations within loci are randomly assigned, and are conserved across loci and individuals. At present, only the Jukes-Cantor model of sequence evolution is implemented.

Bioinformatic pre-processing of reads

Three steps are needed before downstream analysis of the output of ddRADSeqTools with a given RAD-seq analysis pipeline: (1) quantification and removal of PCR duplicates; (2) demultiplexing of reads by individual; and (3) trimming of raw reads.

When using the DBR strategy (Schweyen *et al.* 2014; Tin *et al.* 2015), PCR duplicates can be quantified and removed with *pcrdupremoval.py*. The output of this program generates statistics files reporting the number of total and duplicated reads per locus and individual. This program can also be run for scenarios that do not use the DBR strategy to obtain the percentage of missing data by individual and locus.

189 Reads need to be demultiplexed by individual, in order to build individual genotypes, and to
190 check for the presence of paralogous loci (see Mastretta-Yanes *et al.* 2015). Joint raw reads are
191 demultiplexed by *indsdemultiplexing.py* to obtain separate individual read files.

192 The adapters, primers, indexes and DBRs are removed from raw reads in order to use trimmed
193 reads for alignment and variant calling. The program *readstrim.py* removes the adapters and other
194 sequences from raw reads for the correct alignment of reads and variant calling.

195 The output files of this work-flow are ready to be submitted to alignment utilities, such as
196 BWA (Li & Durbin 2009), or to RADseq analysis pipelines, such as Stacks (Catchen *et al.* 2011) or
197 Pyrad (Eaton 2014), that can provide the number of *in silico* polymorphic loci.

198

199 Validation of correct program operability

200 Four experiments were conducted to validate the correct operability of ddRADseqTools programs, as
201 well as the reliability of the resulting outputs. We wrote specific Bash scripts, modifying the
202 parameters of the program for each validation test (Table 2).

203 Validation test A performed a double digestion of three benchmark genomes with
204 *rsitesearch.py*, each with a different enzyme combination, and a simulated size selection step. The
205 three genomes have contrasting size and degree of complexity, sampled from the kingdoms of Fungi
206 (*Saccharomyces cerevisiae*, 14 chromosomes, small size = 12Mbp, Engel *et al.* 2014), Animalia
207 (*Homo sapiens*, 23 chromosomes, medium size = 3 Gbp, Venter *et al.* 2001), and Plantae (*Pinus taeda*,
208 12 chromosomes, large size = 20 Gbp, Neale *et al.* 2014). The Bash script *simulation-genome.sh*
209 included in the software package has all the instructions to perform this test.

210 Validation test B used *simddradseq.py* to simulate read files from a ddRADseq experiment
211 for 48 individuals of *S. cerevisiae*, under different scenarios for the number of reads to generate
212 (*readsnum*, an indirect estimate of coverage). Three iterations were run for an expected coverage of
213 2x, 4x, 8x, and 16x, respectively. A moderate variation of coverage was simulated setting the

parameters *minreadvar* to 0.8 and *maxreadvar* to 1.2. For each scenario, the mean coverage and the variance for 48 individuals of *S. cerevisiae* and the high and low confidence intervals ($\alpha = 0.5$) were plotted across all loci to test for a correct simulation of unequal coverage among loci and individuals. The Bash script *simulation-unequal-coverage.sh* included in the software package has all the instructions to perform this test.

Validation test C analysed the effect of modifying the theoretical probability of PCR duplicates and the effect of the GC content of the fragments on the number of reads generated for 48 individuals of *S. cerevisiae*, with 4x and 8x coverage. The program *simddradseq.py* generated reads for a range of values for both the probability of loci bearing PCR duplicates (*pcrdupprob* = 0.0-0.9), and a weight factor that multiplies the GC content of a locus (*gcfactor* = 0.0-0.5), to randomize the number of PCR duplicates per locus and individual. To simulate the number of copies per locus with PCR duplicates, we selected a multinomial distribution for a range between one and ten copies. For this range, a vector of probabilities that decreased monotonically was defined to sample the actual number of PCR duplicates. The program *pcrdupremoval.py* quantified and removed the PCR duplicates. The Bash script *simulation-gcfactor.sh* included in the software package has all the instructions to perform this test.

Validation test D was used to check the correct generation of mutations according to a range of user-defined probabilities (0.001-0.1) for 48 samples of *S. cerevisiae*. In this test the programs *rsitesearch.py*, *simddradseq.py*, *pcrdupremoval.py*, and *indsdemultiplexing.py* were run. Statistics of mutated and not-mutated fragments for each individual were calculated based in the information of reads collected in the read headers, and stored in a CSV file. The resulting reads were mapped back to the *S. cerevisiae* reference genome with BWA (Li & Durbin 2009), and performed a variant calling analysis to test for a correct generation of SNP and indel mutations. Besides ddRADSeqTools and BWA, the Bash script *simulation-mutations_polymorphicloci.sh*, included in the software package, used samtools (Li *et al.* 2009), bedtools (Quinlan & Hall 2009), and vcftools (Danecek *et al.* 2011). The output files of alignment and variant calling analyses are returned in SAM, BAM, BED and VCF

format that can be visualized with a genome browser, for instance the Integrative Genome Viewer IGV (Robinson et al. 2011), and are used to compute the percentage of polymorphic loci. The Bash script *simulation-mutations_polymorphicloci.sh* included in the software package contains all the instructions to perform this test.

Correspondence of *in silico* and *in vitro* parameters

The correspondence of parameters from *in vitro* ddRADseq experiments in yeast -*S. cerevisiae*- (Tin *et al.* 2015), ant -*Wasmannia auropunctata*- (Tin *et al.* 2015), viper -*Vipera* sp.- (Zinenko *et al.* 2016), and oilseed rape -*Brassica napus*- (Wu *et al.* 2016) with input settings optimized through a series of runs of ddRADSeqTools were assessed in order to provide some guidance for running the software with reliable parameters. Experiments were selected to cover different features of ddRADseq experiments that have been parameterized in ddRADseqTools, such as enzyme pair combination, range of selected fragment size, type of reads, type of library, length of insert, and number of polymorphic loci (see the specific parameters for each experiment in Table 6).

In all ddRADseq simulations, the total number of loci for the selected insert size and enzyme pair combination, and the percentage of missing data were computed. In order to calculate the number of polymorphic loci, the simulated reads were mapped back to the corresponding reference genomes with BWA (Li & Durbin 2009), and a variant calling analysis was performed. The experiments for *S. cerevisiae* and *W. auropunctata* (Tin *et al.* 2015) adopted a DBR strategy, allowing the comparison between the percentages of experimental and simulated PCR duplicates. The Bash scripts *simulation-pipeline-Scerevisiae-se.sh*, *simulation-pipeline-Wauropunctata-pe.sh, simulation-pipeline-Vberus-se.sh* and *simulation-pipeline-Bnapus-pe.sh* included in the software package have all the instructions to perform the simulations above.

Computational efficiency of ddRADSeqTools

The computational efficiency of the programs that form ddRADSeqTools was assessed with the Bash script *simulation-performance.sh* included in the software package (see the settings of ddRADseqTools to perform this test in Table S1, Supporting information II). In this script, the programs *rsitesearch.py*, *simddradseq.py*, *pcrdupremoval.py*, *indsdemultiplexing.py*, and *readstrim.py* were run repeatedly in order to measure the elapsed real time used by the program, the total number of CPU-seconds used by the system on behalf of the process, the total number of CPU-seconds that the process used directly, and the maximum resident set size of the process during its lifetime. The analysis was run in a computer with Bio-Linux 8 OS. The main features of the computer were Intel Core i5-4200U 1.6 GHz with Turbo Boost up to 2.g GHz; RAM 8 GB; 5400 rpm disk.

Comparison with other *in silico* tools

The number of fragments obtained in validation test A for *S. cerevisiae*, *H. sapiens* and *P. taeda* were compared to the results of analogous simulations performed with simRAD (Lepais & Weir 2014) and the Digital_RADs.py program of BU-RAD-seq (DaCosta & Sorenson 2014) for the same benchmark genomes and enzyme pair combinations. The computational efficiency of ddRADseqTools at *rsitesearch.py* was also compared to the performance of simRAD and BU-RAD-seq.

Results and Discussion

Validation of correct program operability

Validation test A: double digestion and generation of fragments

The summary statistics produced by *rsitesearch.py* for the total number of fragments, and the number of fragments whose size is between the selected size interval for the benchmark genomes and the enzyme pair combinations EcoRI-MseI, PstI-MseI and SbfI-MseI are shown in Table 3. The success and cost-efficiency of a ddRADseq experiment largely depends on the selection of the enzyme pair combination, which can be assessed *in silico* with ddRADSeqTools. The effect of the double digestion

290 with different combinations of enzymes varied depending on the genome of choice. Since the number
291 of reads is a function of the number of fragments multiplied by the coverage and the number of
292 individuals, the enzyme pair chosen in a ddRADseq experiment must provide a tractable number of
293 fragments; that is, there must be a balance between the number of fragments, the total number of reads
294 and the number of individuals to obtain an optimal coverage and a low percentage of missing data. A
295 more detailed graphical representation of the distribution of the resulting fragments by 25 nucleotide
296 size intervals is shown in Figures S1-S3 (Supporting information I). The restriction enzymes marked
297 in bold in Table 2 are considered to provide the optimal number of loci to obtain sufficient coverage
298 across loci and individuals with a reasonable number of reads per experiment.

299

300 *Validation test B: unequal coverage among loci and individuals*

301 This test validated the way ddRADseqTools simulates unequal coverage among loci and individuals
302 with *simddradseq.py*. Figure 3 shows the mean number of reads generated by loci across individuals,
303 and the corresponding low and high confidence intervals for coverage values of 2x, 4x, 8x and 16x.
304 The mean number of reads by locus and individuals oscillated around the expected coverage in all four
305 scenarios, consistently with the *minreadvar* and *maxreadvar* input parameters (0.8 and 1.2
306 respectively). The high and low confidence intervals showed different values for each locus,
307 demonstrating that different coverage was achieved for each individual at each locus.

308

309 *Validation test C: quantification and removal of PCR duplicates*

310 This test performed an in-depth analysis of the effect of the probability of loci bearing PCR duplicates
311 on the number of reads. Table 4 shows the percentage of removed reads, and the coverage deviation
312 for each PCR duplicate probability and coverage (4x and 8x) in *S. cerevisiae*. The results demonstrate
313 the correct operability of *simddradseq.py* and *pcrdupremoval.py*.

The number of removed reads (i.e. the number of duplicate reads) was proportional to the probability of loci bearing PCR duplicates (*pcrdupprob*), and the values were independent of the depth of coverage. The coverage deviation was proportional to both the probability of loci bearing PCR duplicates and the coverage depth. Decreasing coverage, and percentage of loci with missing data became more important as PCR duplicates increased.

The low values scored for the standard deviations of the percentage of removed reads and loci with missing data, respectively, indicate the correct simulation of duplicate reads in relation to variation in the *gcfactor* parameter. Due to an artefact derived from the random generation of the DBR sequences, some duplicate reads were produced when the probability of loci bearing PCR duplicates was 0.0. These duplicate reads occurred also when the probability of loci bearing PCR duplicates was > 0.0, and there is no way to distinguish between real duplicates or artefacts. In any case, the number of duplicate reads generated randomly was negligible when the probability of PCR duplicates was > 0.0.

Validation test D: checking the mutation patterns

The results for this test confirmed a correct generation of mutated reads by the program. After the removal of PCR duplicates and demultiplexing, fragments are annotated with information about the chromosome or scaffold and strand where they belong, and their start and end positions. Reads are annotated with the fragment from where they derived. Files in VCF format allow for the quantification of mutations (SNPs or indels) identified by chromosome or scaffold, and by their coordinates within the genome. The percentage of mutated reads matches the user-defined probabilities (Table 5), confirming that mutations were correctly generated by the program. Also, the number of polymorphic loci calculated after aligning to the reference genome was the expected for each *mutprob* value.

Correspondence of *in silico* and *in vitro* parameters

The results obtained for *in vitro* experiments could be achieved *in silico* setting standard parameters as options in ddRADseqTools. Table 6 shows the correspondence between *in vitro* parameters and input settings for ddRADseqTools. In all cases, the selected enzyme pair combination, the size of the selected fragments to sequence, and the high total number of simulated reads resulted in a null percentage of loci with missing data by individual, suggesting that the correct combination of parameters was selected for the *in vitro* experiments. The deviance between *in silico* generated and empirical number of polymorphic loci was 8% for *S. cerevisiae* and *W. auropunctata*, 15% for *Vipera* sp., and 33% for *B. napus*. The optimal mutation probability depends on the life cycle and mutation rate of the focal organism. While the mutation probability parameter was set to 0.15 for *S. cerevisiae*, a much lower mutation probability was used in the case of *W. auropunctata* (mutprob=0.015), and intermediate values were used for *Vipera* sp. (mutprob=0.1). In the case of *B. napus*, we used a higher mutation probability than expected for the species (mutprob=0.2) to highlight the discordance in terms of the number of polymorphic loci with the results scored *in vitro*.

The experiments of Tin *et al.* (2015) on *S. cerevisiae* and *W. auropunctata* using the DBR approach showed a high percentage of PCR duplicates (48-69% and 31-70%, respectively), that could be obtained *in silico* setting the probability of PCR duplicates to 0.4 and 0.5 respectively, and the GC content factor to 0.2. Schweyen *et al.* (2014), adopting the same DBR strategy, reported a smaller range of PCR duplicates in freshwater invertebrates (12-44%) that could be achieved *in silico* by setting pcrdupprob to 0.2-0.3. Accordingly, in terms of experimental design, all ddRADseqTools settings can be explored to set more conservative or relaxed scenarios and aid in the selection of optimal *in vitro* parameters to simultaneously optimise for limiting missing data and experimental cost.

Computational efficiency of ddRADSeqTools

The programs included in ddRADSeqTools are computationally efficient, and do not require expensive computer infrastructure to be functional. Table 7 shows the performance of the different

programs of ddRADSeqTools after running the script *simulation-performance.sh*, in terms of elapsed real time used by the program, CPU usage, and memory consumption.

The program *rsitesearch.py* needed the highest amount of memory: approximately 61 MiB were required for *S. cerevisiae*; more than 4 GiB for *H. sapiens*; and less than 220 MiB for *P. taeda*. The way the reference genome files are structured also has an impact on the performance of *rsitesearch.py*. Although the genome of *P. taeda* is much larger than that of *H. sapiens*, memory requirements for the scaffolded *P. taeda* genome were lower than for *H. sapiens* that presented a more complex structural arrangement with chromosomes. The elapsed time depended both on the genome size and on the number of fragments obtained (Table 7).

The program *simddradseq.py* had very low memory requirements: below 23 MiB for the three reference genomes analysed, and the elapsed time was proportional to the number of reads (Table 7). The maximum elapsed time recorded was less than 39 min for *P. taeda* with 16x coverage (2 400 000 simulated reads). The performance of the program *pcrdupremoval.py* depended largely on the number of records in the input and the output files: for a fixed size of the input file, the execution time was directly proportional to the value of the *pcrdupprob* parameter. The maximum elapsed time recorded was approximately 2 hr and 57 min for *P. taeda* with 16x coverage, and a probability of 0.2 for loci bearing PCR duplicates.

The program *insdemultiplexing.py* consistently had a memory requirement of approximately 10 MiB. Again, the elapsed time depended on the records in the input file. For a fixed coverage, higher *prcrdupprob* values implied less number of reads in the files where the PCR duplicates were already removed. The maximum elapsed time recorded was 21 min and 11 s for *P. taeda* with 16x coverage, and a probability of 0.2 for loci bearing PCR duplicates. The program *readstrim.py* was also very efficient. The memory consumption was below 9 MiB, and the mean elapsed real time was 5 min and 12 s for *P. taeda* with 16x coverage, and a probability of 0.2 for loci bearing PCR duplicates.

390 Comparison with other *in silico* tools

391 On the one hand, the program *rsitesearch.py* showed good performance in comparison to both the R
392 package SimRAD (Lepais & Weir 2014) and Digital_RADs.py of BU-RAD-seq (DaCosta & Sorenson
393 2014). The number of fragments of different sizes sampled from the benchmark genomes for different
394 enzyme pair combinations varied only slightly among the three applications (Table 8), probably due to
395 differences in size selection algorithms or in the treatment of N's in the genomes. Particularly,
396 *rsitesearch.py* was computationally efficient when executed against large or complex genomes. The
397 software simRRLs (Eaton 2014) is a good alternative for phylogenetic ddRADseq studies, because it
398 builds read files conditioned upon an input tree topology based on coalescence. simRRLs generates
399 random sequences for several modifications of the RADseq methodology, including ddRADseq, and
400 also incorporates some sources of error to the read simulation procedure, such as allele dropout or low
401 coverage. However, unlike ddRADseqTools, it does not generate reads from a reference genome and
402 the current version does not handle PCR duplicates.

403

404 Limitations of ddRADseqTools

405 The current version of ddRADseqTools presents some limitations: (1) when ddRADseqTools is run
406 without a reference genome, it only provides randomly generated reads, that can be used to estimate
407 computational times in further ddRADseq bioinformatic pipelines, rather than provide specific
408 information about the design of the experiment for the focal species; (2) the mutation model currently
409 implemented in ddRADSeqTools does not consider the possibility of simulating individuals with
410 varying degree of relatedness; (3) mutations are incorporated only according to the Jukes-Cantor
411 model of sequence evolution; (4) mismatches are not admitted in the demultiplexing process. This
412 limitation is not important when reads are generated *in silico*, as in the examples presented here, but
413 the current version of *pcrdupremoval.py* and *indsdemultipling.py* should be used with caution with
414 experimental ddRADseq data; and (5) paralagous sequences are not parameterized. If genomes with a
415 high content of repetitive regions (e.g. *P. taeda*) are used as a reference, some paralogous fragments

416 will be generated, but this is a feature not controlled by the user. However, paralogous sequences can
417 be identified following Mastretta-Yanes *et al.* (2015). When reads are generated at random with
418 *fragsgeneration.py*, paralogous sequences are not generated. Subsequent versions of the software will
419 address these limitations.

420

421 **Conclusions**

422 ddRADseqTools is a flexible application to facilitate the *in silico* design of ddRADseq experiments.
423 The software is adaptable to a broad range of conditions, such as the construction of modified
424 ddRADseq libraries using adapters with either one or two indexes, and degenerate base regions
425 (DBRs) to quantify PCR duplicates. Simulations with ddRADseqTools may be used to estimate an
426 optimal enzyme pair combination and size range for sequenced fragments, and to simulate scenarios to
427 predict the impact of PCR duplicates or allele dropout on coverage and missing data. It performs the
428 initial bioinformatic pre-processing of reads, so *in silico* reads can then be downstreamed to
429 ddRADseq analysis pipelines to estimate the number of polymorphic loci or to perform specific tests
430 with simulated data. The software runs efficiently in computers with Linux/Unix, Mac OS or
431 Microsoft Windows, and standard CPU and RAM configuration.

432

433 **Acknowledgements**

434 We would like to thank N. Álvarez and A. Mastretta-Yanes for fruitful discussions and
435 support. This work was supported in part by Spanish MINECO grant CGL2013-42589-P co-financed
436 by FEDER, and FPI studentship BES-2014-067868.

437

438 **References**

439 Baird NA, Etter PD, Atwood TS *et al.* (2008) Rapid SNP discovery and genetic mapping using
440 sequenced RAD markers. *PLos ONE*, **3**, e3376.

Blainey P, Krzywinski M, Altman N (2014) Points of significance: replication. *Nature Methods*, **11**, 879-880.

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.

Catchen J, Hohenlone PA, Bassham S, Amores A, Cresko WA (2013) Stacks: an analysis tool set for population genomics. *Molecular Ecology*, **22**, 3124-40.

Danecek P, Auton A, Abecasis G *et al.* (2011) The variant call format and VCFtools. *Bioinformatics*, **27**, 2156-8.

DaCosta JM, Sorenson MD (2014) Amplification biases and consistent recovery of loci in a double-digest RAD-seq protocol. *PLoS ONE*, **9**, e106713.

Davey JW, Blaxter ML (2010) RADSeq: next-generation population genetics. *Briefing in Functional Genomics*, **9**, 416-423.

Davey JW, Cezard T, Fuentes-Utrilla P, Eland C, Gharbi K, Blaxter ML (2013) Special features of RAD sequencing data: implications for genotyping. *Molecular Ecology*, **22**, 3151-3164.

Davey JW, Hohenlohe PA, Etter PD *et al*. (2011) Genome-wide genetic marker discovery and genotyping using next-generation sequencing. *Nature Reviews Genetics*, **12**, 499-510.

Eaton DAR (2014) PyRAD: assembly of *de novo* RADseq loci for phylogenetic analyses. *Bioinformatics*, **30**, 1844-1849.

Etter PD, Bassham S, Hohenlohe PA, Johnson EA, Cresko WA (2011) SNP discovery and genotyping for evolutionary genetics using RAD sequencing. *Methods in Molecular Biology*, **772**, 157-178.

Engel SR, Dietrich FS, Fisk DG *et al*. (2014) The reference genome sequence of *Saccharomyces cerevisiae*: then and now. *G3: Genes, Genomes, Genetics*, **4**, 389-398.

Gautier M, Gharbi K, Cezard T *et al.* (2013) The effect of RAD allele dropout on the estimation of genetic variation within and between populations. *Molecular Ecology*, **22**, 3165-3178.

Lepais O, Weir JT (2014) SimRAD: an R package for simulation-based prediction of the number of loci expected in RADseq and similar genotyping by sequencing approaches. *Molecular Ecology Resources*, **14**, 1314-1321.

Li H, Durbin R (2009) Fast and accurate short read alignment with Burrows–Wheeler Transform. *Bioinformatics*, **25**, 1754-1760.

Li H, Handsaker B, Wysoker A *et al*. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078-2079.

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.

Mastretta-Yanes A, Zamudio S, Jorgensen TH *et al.* (2014). Gene duplication, population genomics, and species-level differentiation within a tropical mountain shrub. *Genome Biology and Evolution*, **6**, 2611-2624.

Neale DB, Wegrzyn JL, Stevens KA *et al.* (2014) Decoding the massive genome of loblolly pine using haploid DNA and novel assembly strategies. *Genome Biology*, **15**, R59.

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.

Quinlan AR, Hall IM (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, **26**, 841-842.

Recknagel H, Elmer KR, Meyer A. 2013. A hybrid genetic linkage map of two ecologically and morphologically divergent Midas cichlid fishes (*Amphilophus* spp.) obtained by massively parallel DNA sequencing (ddRADSeq). *G3 (Bethesda)*, **3**, 65-74.

Robinson JT, Thorvaldsdóttir H, Winckler W *et al*. (2011) Integrative genomics viewer. *Nature Biotechnology*, **29**, 24-26.

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.

Venter JC, Adams MD, Myers EW *et al.* (2001) The sequence of the human genome. *Science*, **291**, 1304-1351.

Wu Z, Wang B, Chen X *et al.* (2016) Evaluation of linkage disequilibrium pattern and association study on seed oil content in *Brassica napus* using ddRAD sequencing. *PLoS ONE* 11(1), e0146383.

Zinenko O, Sovic M, Joger U, Gibbs HL (2016) Hybrid origin of European Vipers (*Vipera magnifica* and *Vipera orlovi*) from the Caucasus determined using genomic scale DNA markers. *BMC Evolutionary Biology*,16(1), 76.

Data Accessibility

ddRADseqTools along with its manual, and the validation scripts are available from the software repository GitHub (https://github.com/GGFHF/ddRADseqTools).

Author Contributions

ULH, FMM and BCE conceived the ideas. FMM programmed the software. FMM, ULH and VGO performed the tests to validate the software. ULH wrote the manuscript. All authors commented on and approved the final version of the manuscript.

Supporting information

Supporting information I:

Table S1. Values of the main options set in the runs of each ddRADseq program in *simulation-performance.sh*.

Supporting information II:

Figure S1. Distribution of fragments after a double digest of *S. cerevisiae* genome with EcoRI-MseI, PstI-MseI and SbfI-MseI enzyme pair combinations drawn by *rsitesearch.py*.

Figure S2. Distribution of fragments after a double digest of *H. sapiens* genome with EcoRI-MseI, PstI-MseI and SbfI-MseI enzyme pair combinations drawn by *rsitesearch.py*.

Figure S3. Distribution of fragments after a double digest of *P. taeda* genome with EcoRI-MseI, PstI-MseI and SbfI-MseI enzyme pair combinations drawn by *rsitesearch.py*.

524 **Tables and Figures**

525 **Table 1.** Parallelism between the *in vitro* and *in silico* initial steps in a ddRADSeq experiment. The
526 programs that perform each step in ddRADseqTools are indicated. The output of ddRADSeqTools is
527 further downstreamed to an alignment or de novo assembly RADseq pipeline.

528

***In vitro* experiments**	***In silico* experiments**	**ddRADseqTools program**
Library construction	*In silico* fragments generation	*rsitesearch.py* (w/genome) *fragsgeneration.py* (random)
High-Throughput Sequencing	Generation of reads	*simddradseq.py*
Bioinformatics pre-processing of reads		
Quantification and removal of PCR duplicates		*pcrdupremoval.py*
Demultiplexing of individuals		*indsdemultiplexing.py*
Trimming of raw reads		*readstrim.py*

529

530

Table 2. Parameters used in tests A-D to validate the correct operability of ddRADseqTools.

Options	***Test A***	***Test B***	***Test C***	***Test D***
enzyme1	EcoRI, SbfI & PstI	EcoRI	EcoRI	EcoRI
enzyme2	MseI	MseI	MseI	MseI
fragstinterval	25	25	25	25
genfile	*S. cerevisiae*† *H. sapiens*‡ *P. taeda*#	*S. cerevisiae*†	*S. cerevisiae*†	*S. cerevisiae*†
minfragsize	101 (*S. cerevisiase*) 201 (*H. sapiens* and *P. taeda*)	101	101	101
maxfragsize	300	300	300	300
individualsfile	-	file with 48 individuals	file with 48 individuals	file with 48 individuals
index1len	-	6	6	6
index2len	-	6	6	6
dbrlen	-	4	4	4
format	-	FASTQ	FASTQ	FASTQ
fragsfile	-	Output of *rsitesearch.py*	Output of *rsitesearch.py*	Output of *rsitesearch.py*
readtype	-	PE	PE	PE
technique	-	IND1_IND2	IND1_IND2_DBR	IND1_IND2_DBR
Readsnum (coverage)	-	300 000 (2x) 600 000 (4x) 1 200 000 (8x) 2 400 000 (16x)	600 000 (4x) 1 200 000 (8x)	300 000
locinum	-	3000	3000	3000
insertlen	-	100	100	100
mutprob	-	0.2	0.2	0.001, 0.010, 0.020, 0.030, 0.040, 0.050, 0.060, 0.070, 0.080, 0.090, 0.100
indelprob	-	0.1	0.1	0.1
locusmaxmut	-	1	1	1
maxindelsize	-	10	10	10
maxreadvar	-	1.2	1.2	1.2
minreadvar	-	0.8	0.8	0.8
dropout	-	0.0	0.0	0.0
pcrdistribution	-	-	MULTINOMIAL	MULTINOMIAL
multiparam	-	-	0.167, 0.152, 0.136, 0.121, 0.106, 0.091, 0.076, 0.061, 0.045, 0.030, 0.015	0.167, 0.152, 0.136, 0.121, 0.106, 0.091, 0.076, 0.061, 0.045, 0.030, 0.015
pcrdupprob	-	0.0	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9	0.2
gcfactor	-	-	0.0, 0.1, 0.2, 0.3, 0.4, 0.5	0.2

† ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF_000146045.2_R64/GCF_000146045.2_R64_genomic.fna.gz
‡ ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF_000001405.29_GRCh38.p3/GCF_000001405.29_GRCh38.p3_genomic.fna.gz
# http://dendrome.ucdavis.edu/ftp/Genome_Data/genome/pinerefseq/Pita/v1.01/ptaeda.v1.01.scaffolds.fasta.gz

Table 3. Fragments generated by restriction endonucleases for three reference genomes (*S. cerevisiae*, *H. sapiens*, and *P. taeda*). The optimal enzyme combination inferred from the number of fragments generated for the selected size interval is indicated in bold.

S. cerevisiae

Enzymes	Total fragments	Fragments w/ size 101-300 nt
EcoRI - MseI	8,176	**3,103**
PstI - MseI	4,623	1,853
SbfI - MseI	188	70

H. sapiens

Enzymes	Total fragments	Fragments w/ size 201-300 nt
EcoRI - MseI	1,629,978	203,735
PstI - MseI	2,236,406	331,344
SbfI - MseI	156,140	**21,016**

P. taeda

Enzymes	Total fragments	Fragments w/ size 201-300 nt
EcoRI - MseI	11,459,733	1,353,309
PstI - MseI	4,784,215	621,933
SbfI - MseI	215,211	**26,532**

Table 4. Percentage of removed reads, coverage deviation and percentage of loci with missing data for a range of theoretical *pcrdupprob* values (0.0-0.9), iterated five times each (gcfactor = 0.0-0.5). Mean and standard deviation (in brackets) of iterations are shown. Data for 48 *S. cerevisiae* individuals at 4x and 8x coverage simulated in test C.

	4x			8x		
pcrdupprob	**% removed reads**	**Coverage deviation**	**% of loci with missing data**	**% of removed reads**	**Coverage deviation**	**% of loci with missing data**
0.0	0.72 (0.02)	-0.03 (0.00)	1.50 (0.52)	1.42 (0.01)	-0.11 (0.01)	0.00 (0.00)
0.1	9.09 (0.79)	-0.38 (0.04)	5.67 (0.98)	9.95 (1.22)	-0.82 (0.10)	1.67 (0.65)
0.2	16.31 (1.09)	-0.67 (0.04)	9.00 (1.21)	16.82 (0.58)	-1.40 (0.05)	3.00 (1.04)
0.3	23.77 (0.44)	-0.99 (0.02)	12.50 (1.31)	24.70 (0.70)	-2.06 (0.05)	4.58 (0.90)
0.4	31.64 (0.46)	-1.32 (0.02)	16.33 (1.50)	32.12 (0.23)	-2.67 (0.03)	6.00 (1.04)
0.5	39.81 (0.71)	-1.66 (0.03)	20.42 (1.56)	39.90 (0.94)	-3.33 (0.07)	7.17 (1.11)
0.6	46.58 (0.58)	-1.94 (0.03)	23.33 (1.72)	47.04 (1.21)	-3.92 (0.09)	8.83 (1.27)
0.7	54.45 (0.53)	-2.27 (0.02)	27.17 (1.64)	54.69 (1.05)	-4.55 (0.09)	10.42 (1.44)
0.8	61.82 (0.55)	-2.58 (0.02)	30.75 (1.71)	62.09 (0.45)	-5.17 (0.04)	11.75 (1.54)
0.9	68.92 (0.88)	-2.87 (0.04)	34.00 (2.04)	69.14 (0.47)	-5.76 (0.04)	13.17 (1.47)

549 **Table 5.** Number of total and mutated reads, and of polymorphic loci for 48 individuals of *S.*
550 *cerevisiae* obtained for validation test D with values of *mutbprob*=0.0-0.1. The percentage of mutated
551 reads and polymorphic loci is shown in brackets.

mutprob	Total reads	Mutated reads (%)	Polimorphic loci (%)
0 .001	251 773	552 (0.1)	111 (3.6)
0 .010	253 729	2 513 (1.0)	915 (29.5)
0 .020	250 896	4 973 (2.0)	1507 (48.6)
0 .030	252 112	7 733 (3.0)	1899 (61.2)
0 .040	252 189	9 947 (3.9)	2165 (69.8)
0 .050	252 746	12 705 (5.0)	2335 (75.3)
0 .060	252 835	14 961 (5.9)	2403 (77.4)
0 .070	253 788	17 524 (6.9)	2493 (80.3)
0 .080	251 965	20 087 (8.0)	2547 (82.1)
0 .090	250 839	22 404 (9.0)	2585 (83.31)
0 .100	253 335	25 347 (10.0)	2600 (83.8)

552

553

554 **Table 6.** Correspondence between parameters of *in vitro* ddRADseq experiments and parameters set as options in ddRADseqTools.
555

	Tin *et al.* (2015)		Tin *et al.* 2015		Zinenko *et al.* 2016		Wu *et al.* 2016	
Experiment /ddRADseqTools parameter	**Experiment parameters**	**ddRADseqTools parameters**	**Experiment parameters**	**ddRADseqTools parameters**	**Experiment parameters**	**ddRADseqTools parameters**	**Experiment parameters**	**ddRADseqTools parameters**
Organism / genfile	*Saccharomyces cerevisiae*	*S. cerevisiae*†	*Wasmannia auropunctata*	*W. auropunctata*†	*6 Vipera species*	*Vipera berus*†	*Brassica napus*	*B. napus*†
1st restriction enzyme / enzyme1	EcoRI	EcoRI	EcoRI	EcoRI	EcoRI	EcoRI	SacI	SacI
2nd restriction enzyme / enzyme2	MseI	MseI	MseI	MseI	SbfI	SbfI	MseI	MseI
Lower boundary of size selection /minfragsize	300	87‡	400	187‡	300	230#	270	140‡
Upper boundary of size selection / maxfragsize	700	487‡	500	287‡	450	380#	550	420‡
Number of loci to simulate	-	**4353**	-	**18159**	-	**2351**	-	**110 464**
Number of individuals / content of individuals.txt file	5	5 index sequences	5	5 index sequences	40	40 index sequences	189	189 index sequences
Total number of reads / readsnum	5 629 058 - 4 518 638	5 000 000	4 967 954 - 6 733 656	5 000 000	3 300 000	3 300 000	506 810 000	506 810 000
Read type / readtype	SE	SE	PE	PE	SE	SE	PE	PE
Library type / technique		IND1_DBR		IND1_IND2_DBR		IND1		IND1_IND2
Library type / index1len	Single 7 bp barcode and a DBR of 4 bp	7	Two 7 bp barcodes and a DBR of 4 bp	7	Single index (no DBR)	6	Single index (no DBR)	5
Library type / index2len		0		7		0		5
Library type / dbrlen		4		4		0		0
Read length / insertlen	50	50	25	25	50	50	80	80
Format of reads file / format	.fastq	FASTQ	.fastq	FASTQ	.fastq	FASTQ	.fastq	FASTQ
% of duplicate reads / pcrdupprob	48-69%	0.4 / **51%**	31 - 70%	0.5 / **45%**	-	-	-	-
GC content / gcfactor	-	0.2	-	0.2	-	0.2	-	0.2
Mutation probability / mutprob	-	0.15	-	0.015	-	0.10	-	0.2
Probability of indels / indelprob	-	0.1	-	0.1	-	0,1	-	0.1
Maximum number of mutations by locus / locusmaxmut	-	1	-	1	-	1	-	1
Allele dropout probability /dropout	-	0.05	-	0.015	-	0,05	-	0.0

Upper indel size / maxindelsize	-	10	-	10	-	10	-	10
Lower threshold value for inter-locus coverage variation / minreadvar	-	0.8	-	0.8	-	0,8	-	0.8
Upper threshold value for inter-locus coverage variation / minreadvar	-	1.2	-	1.2	-	1,2	-	1.2
Number of polymorphic loci	2774*	**2998**	2331*	**2151**	1959	**1668**	31 833	**42406**
Average % of missing data by individual	-	**0.0%**	-	**0.0%**	-	**0.01%**	-	**0.0%**

In bold the output of ddRADseqTools.
† Genome assemblies download from NCBI genome database.
‡ Length of both adapters and primer pairs were substracted from actual size because size selection was performed after ligation and before attachment of PCR primers.
# Length of the adaptor was not specified in Zinenko *et al.* (2016). We assumed a length of the adaptor of 70 bp that was substracted form the original size length to perform the simulations.
* Polymorphic loci after PCR duplicates removal.

564

565 **Table 7.** Performance data of the programs *rsitesearch.py*, *simddradseq.py*, *pcrdupremoval.py*, *indsdemultiplexing.py*, and *readstrim.py* collected from
566 a run of *simulation-performance.sh* in a PC with Bio-Linux 8 OS, an Intel Core i5-4200U 1.6 GHz with Turbo Boost up to 2.g GHz processor, RAM of
567 8 GB, and a 5400 rpm disk.

Program	organims	enzyme 1-enzyme2	readsnum	pcrdupprob	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode	Percentage of CPU	maximum resident set size (Kb)
rsitesearch.py	*S. cerevisiae*	EcoRI-MseI	-	-	5.0	0.1	2.4	49%	62 352
		PstI-MseI	-	-	2.3	0.0	2.2	94%	62 316
		SbfI-MseI	-	-	2.0	0.1	1.8	93%	60 688
	H. sapiens	EcoRI-MseI	-	-	421.4	11.9	399.3	97%	4 388 504
		PstI-MseI	-	-	469.8	10.0	457.6	99%	4 391 800
		SbfI-MseI	-	-	324.2	8.2	314.5	99%	4 387 148
	P. taeda	EcoRI-MseI	-	-	2 812.5	19.3	2 773.9	99%	222 840
		PstI-MseI	-	-	2 429.5	15.4	2, 402.5	99%	214 040
		SbfI-MseI	-	-	2 005.7	11.5	1 985.6	99%	205 528
simddradseq.py	*S. cerevisiae*	EcoRI-MseI	300 000	0.2	33.7	1.1	14.1	45%	10 676
				0.6	34.4	1.2	11.8	37%	10 680
			2 400 000	0.2	278.3	8.7	107.9	41%	10 676
				0.6	280.1	9.3	87.5	34%	10 680
	H. sapiens	SbfI-MseI	2 000 000	0.2	228.1	7.4	104.8	49%	20 280
				0.6	227.9	7.6	87.4	41%	20 280
			16 100 000	0.2	1 843.2	61.2	818.2	47%	20 276
				0.6	1 838.4	64.8	671.8	40%	20 284
	P. taeda	SbfI-MseI	2 500 000	0.2	287.3	9.1	129.7	48%	23 324
				0.6	286.8	9.5	108.7	41%	23 176
			20 400 000	0.2	2 333.5	78.0	1 045.2	48%	23 172
				0.6	2 337.6	79.9	839.2	39%	23 168
pcrdupremoval.py	*S. cerevisiae*	EcoRI-MseI	300 000	0.2	157.1	3.4	139.2	90%	453 764
				0.6	141.9	3.2	127.9	92%	441 848
			2 400 000	0.2	1 047.6	26.6	862.3	84%	1 008 240
				0.6	989.1	24.4	833.9	86%	1 008 276
	H. sapiens	SbfI-MseI	2 000 000	0.2	1 267.9	23.2	1 137.7	91%	2 413 072
				0.6	1 127.2	21.4	1 012.2	91%	2 325 848
			16 100 000	0.2	7 738.3	195.1	6 107.0	81%	2 622 616
				0.6	7 360.4	175.9	5 880.7	82%	2 616 196

	P. taeda	SbfI-MseI	2 500 000	0.2	1 629.3	30.4	1 456.4	91%	2 988 600
				0.6	1 492.4	27.1	1 344.6	91%	2 909 216
			20 400 000	0.2	10 606.2	261.6	8 166.9	79%	3 248 108
				0.6	9 560.6	234.3	7 337.3	79%	3 246 888
indsdemultiplexing.py	*S. cerevisiae*	EcoRI-MseI	300 000	0.2	17.1	0.9	5.6	38%	10 440
				0.6	10.9	0.7	3.5	38%	10 436
			2 400 000	0.2	143.4	7.5	41.2	33%	10 440
				0.6	91.9	4.7	27.0	34%	10 428
	H. sapiens	SbfI-MseI	2 000 000	0.2	121.8	6.9	35.6	34%	10 436
				0.6	75.4	3.8	22.1	34%	10 444
			16 100 000	0.2	996.2	56.3	281.0	33%	10 436
				0.6	640.1	37.5	180.5	34%	10 432
	P. taeda	SbfI-MseI	2 500 000	0.2	149.8	8.2	45.1	35%	10 440
				0.6	95.6	5.7	28.8	36%	10 436
			20 400 000	0.2	1 270.9	67.9	351.0	32%	10 436
				0.6	818.4	46.7	230.0	33%	10 440
readstrim.py	*S. cerevisiae*	EcoRI-MseI	300 000	0.2	3.9	0.3	3.3	98%	9 096
				0.6	3.3	0.1	2.9	98%	9 096
			2 400 000	0.2	19.3	1.3	13.7	91%	9 096
				0.6	13.7	0.8	9.6	93%	9 096
	H. sapiens	SbfI-MseI	2 000 000	0.2	15.7	1.0	12.3	94%	9 096
				0.6	10.2	0.8	8.5	96%	9 094
			16 100 000	0.2	237.9	11.3	90.3	45%	9 096
				0.6	160.5	8.0	60.7	47%	9 096
	P. taeda	SbfI-MseI	2 500 000	0.2	18.1	1.2	15.0	94%	9 096
				0.6	12.8	0.9	10.2	94%	9 094
			20 400 000	0.2	311.4	15.2	112.4	43%	9 094
				0.6	208.1	9.9	76.3	45%	9 095

568

569

Table 8. Comparison between *rsitesearch.py* and the R package SimRAD (Lepais & Weir 2014) and Digital_RADs.py of BU-RAD-seq (DaCosta & Sorenson 2014).

S. cerevisiae

	ddRADseqTools - rsitesearch.py					SimRAD (*)					BU-RAD-seq - Digital_RADs.py (*) (**) (***)				
enzymes	total fragments	fragments w/ size 101-300 nt	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode	total fragments	fragments w/ size 101-300 nt	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode	fragments w/ size 1-1,000 nt	fragments w/ size 101-300 nt	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode
EcoRI - MseI	8 176	3 103	4.99	0.09	2.38	8,176	3,048	21.14	0.21	17.94	8 139	3 191	1.30	0.03	0.41
PstI - MseI	4 623	1 853	2.34	0.01	2.19	4,628	1,866	18.32	0.21	18.07	4 590	1 934	0.38	0.02	0.36
SbfI - MseI	188	70	2.01	0.05	1.84	188	70	17.89	0.20	17.66	186	73	0.35	0.01	0.34

H. sapiens

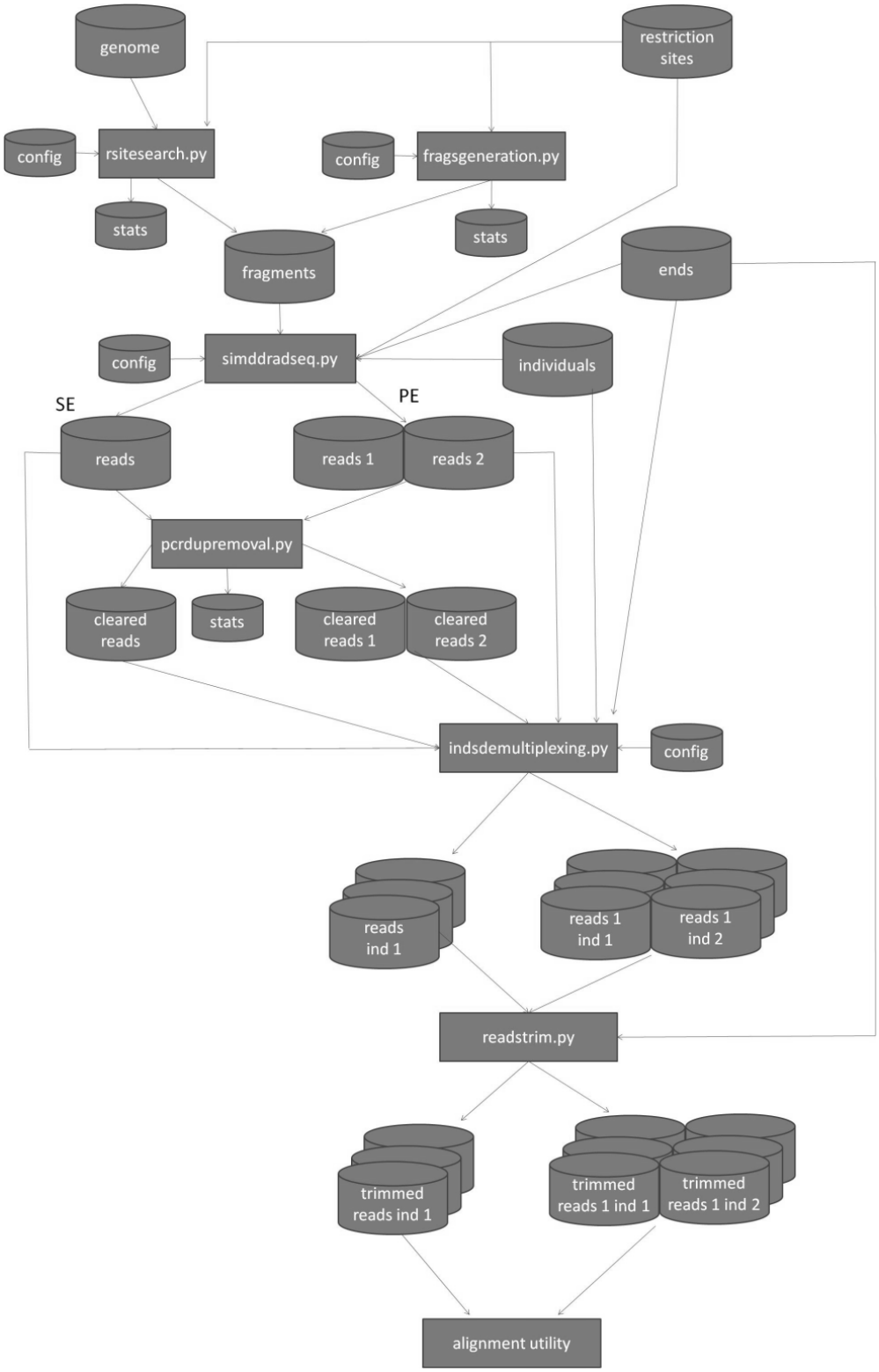
	ddRADseqTools					SimRAD (*)					BU-RAD-seq - Digital_RADs.py (*) (**) (***)				
enzymes	total fragments	fragments w/ size 201-300 nt	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode	total fragments	fragments w/ size 201-300 nt	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode	fragments w/ size 1-1,000 nt	fragments w/ size 201-300 nt	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode
EcoRI - MseI	1 629 978	203 735	421.39	11.90	399.33	(****)	(****)	(****)	(****)	(****)	1 604 730	208 238	233.08	8.89	96.03
PstI - MseI	2 236 406	331 344	469.76	10.03	457.63	(****)	(****)	(****)	(****)	(****)	2 195 695	343 793	180.33	5.66	87.17
SbfI - MseI	156 140	21 016	324.16	8.18	314.54	(****)	(****)	(****)	(****)	(****)	141 656	21 660	175.42	5.37	84.21

P. taeda

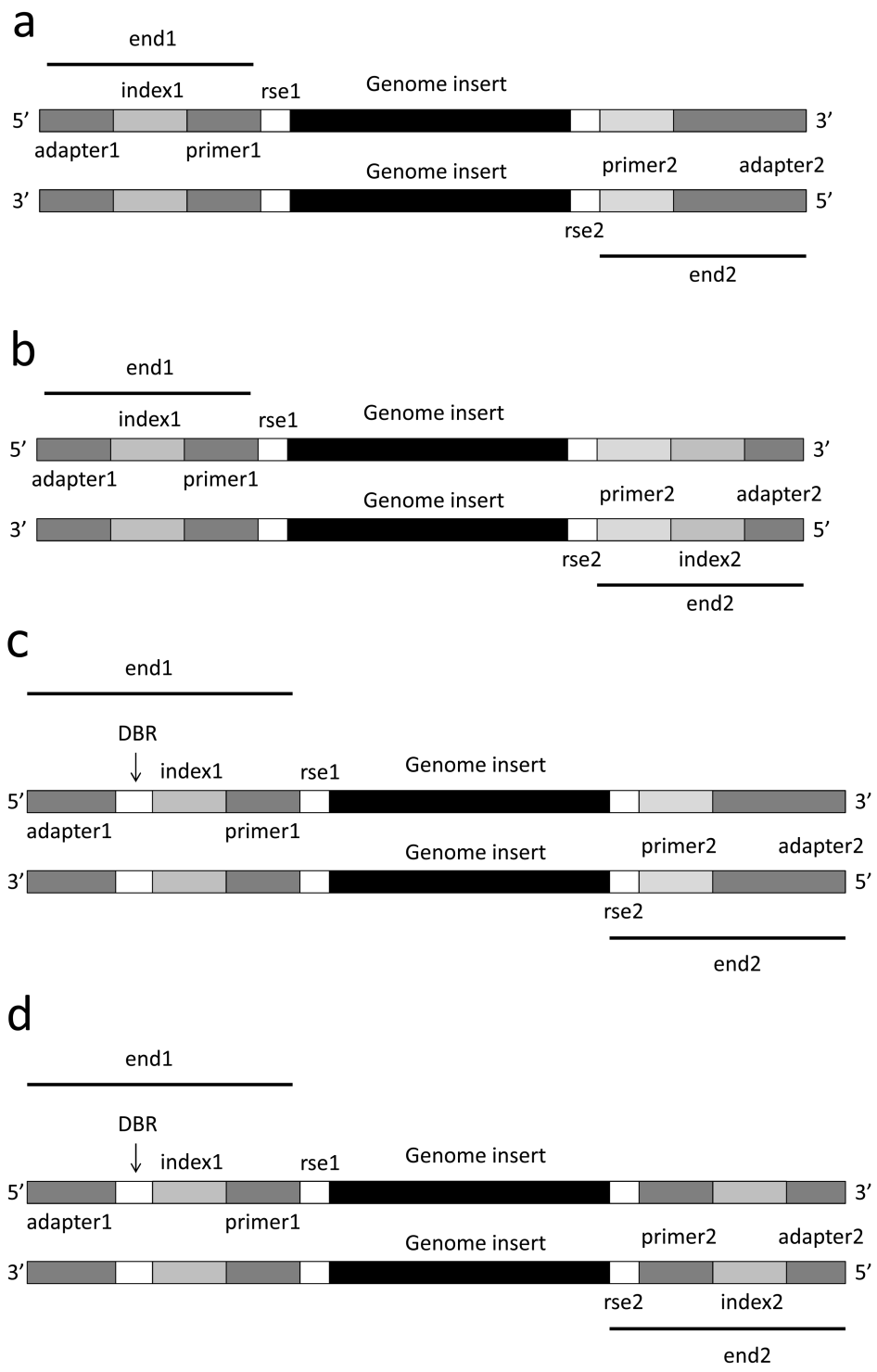
	ddRADseqTools					SimRAD (*)					BU-RAD-seq - Digital_RADs.py (*) (**) (***)				
enzymes	total fragments	fragments w/ size 201-300 nt	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode	total fragments	fragments w/ size 201-300 nt	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode	fragments w/ size 1-1,000 nt	fragments w/ size 201-300 nt	elapsed real time (s)	CPU time (s) in kernel mode	CPU time (s) in user mode
EcoRI - MseI	11 459 733	1 353 309	2 812.50	19.27	2 773.89	(*****)	(*****)	(*****)	(*****)	(*****)	11 181 647	1 377 129	26 937.80	872.16	4,062.31
PstI - MseI	4 784 215	621 933	2,429.52	15.42	2 402.45	(*****)	(*****)	(*****)	(*****)	(*****)	4 590 018	643 991	34 287.74	902.78	4,141.07
SbfI - MseI	215 211	26 532	2,005.67	11.48	1 985.56	(*****)	(*****)	(*****)	(*****)	(*****)	204 438	27 408	68 824.32	955,48	4,336.22

(*) It was necessary to decompress the genome file in a preliminar stage. Elapsed real time: *S. cerevisiae*, 0.14 s; *H. sapiens*, 59.96 s; *P. taeda*, 443,30 s.
(**) It was necessary to convert genome file content to upper case previously. Elapsed real time: *S. cerevisiae*, 0.14 s; *H. sapiens*, 100.81 s; *P. taeda*, 829.36 s.
(***) Further, it was necessary to delete temporal files. For *P. taeda*, 14 412 988 temporal files were generated and their deletion took several hours.
(****) Error in ref.DNAseq (result would exceed 2^31-1 bytes). (*****) Computer crashed.

Figure 1. Overview of the work-flow of ddRADSeqTools. The input and output files for each application are indicated. The last step in the work-flow produces an input for pipelines of genome alignment or of *de novo* assembly.



582 **Figure 2.** Scheme of index and/or DBR positions in the adapters of the four types of library
583 implemented in ddRADseqTools. (a) a single index in *Adapter 1*; (b) one index in *Adapter 1* and
584 another index in *Adapter 2*; (c) a single index and a DBR in *Adapter 1*; and (d) one index in *Adapter 1,*
585 another index in *Adapter 2*, and a DBR.
586



587
588
589

Figure 3: Mean actual coverage by locus across individuals for 2x, 4x, 8x and 16x simulations for 48 individuals of *S. cerevisiae* in validation test B. The high and low confidence intervals for $\alpha = 0.05$ are shown in grey.