Additional instructions for data simulations and analyses of simulated and real data

This document provides additional information, which is necessary to reproduce the analyses that we describe in the manuscript.

We used nine additional Perl scripts to simulate data, to evaluate the results obtained with GIbPSs and pyRAD and to perform analyses with pyRAD. PyRAD is available at https://github.com/dereneaton/pyrad. Below, we describe in detail how we used the additional Perl scripts. Their names are sim_dat_07.pl, subsamp_e_N_01.pl, sim_fastq_02.pl, comp_simres_06.pl, pR_comp_simres_05.pl, comp_ind_gt_03.pl, remtrim_02.pl, pRdespac_01.pl, and pRalleles_06.pl.

The scripts sim_dat_07.pl, simfastq_02.pl, and subsamp_e_N_01.pl require the Perl module List::Util. The script sim_fastq_02.pl requires the Perl modules IO::File and IO::Uncompress::Gunzip. The script comp_simres_06.pl requires the Perl module Math::Round. GIbPSs also requires these modules. We used remtrim_02.pl and pR_comp_simres_05.pl with Perl 5.18.2 under Ubuntu 14.04 LTS and all other scripts with Strawberry Perl 5.16.2 under Windows 7.

Creation of simulated datasets, series A-D

We simulated four series of datasets, A, B, C, and D. In the following, we describe how to use sim_dat_07.pl and subsamp_e_N_01.pl to repeat the simulations.

Series A:

Create a directory, e.g. C:\sim_data\seriesA and place copies of the Perl scripts sim_dat_07.pl and subsamp_e_N_01.pl in that directory.

Call sim_dat_07.pl as follows:

```
sim_dat_07.pl -nI 100 -nL 1000 -nR 100 -Lsl 120 -Ld 20 -Rl 80 -nM 0
-e 0 -N 0 -i 0 -d 0 -rc 1 -o C:\sim_data\seriesA\baseR100e0
```

The program creates a directory C:\sim_data\seriesA\baseR100e0 and produces outfiles therein. It is important to provide a full path to the outfile directory with flag -o as above. With the above settings, the program simulates a dataset with 100 individuals, 1000 monomorphic loci, and 100 read pairs without sequencing error from each locus and individual. Loci have a length of 120. Reads have a length of 80. The program produces one fastq file per individual with one sequence per read pair. The sequence consists of the forward read followed by the reverse complement of the reverse read. This format corresponds to edited sequence files produced by GIbPSs in analyses of paired-end reads. Forward reads stem from one or the other strand of the locus with probability 0.5. The program produces several additional files. Some of them are needed by subsequent programs. File inds_infiles.txt contains individual IDs and corresponding fastq file names. File sim_dat_rep.txt contains the settings used by the program and a random number seed (rs). You can provide this random number seed with the additional flag -rs to reproduce the same outfiles a second time. Names of the other outfiles are intact_reads.txt, ori_gt.txt, ori_popall.txt, ori_poploc.txt, and popall.fas.

The Perl script subsamp_e_N_01.pl subsamples the sequence data created by sim_dat_07.pl and introduces sequencing error. Call the program as follows to create a sequence dataset with 40 read pairs per locus and individual and a sequencing error probability of 0.001:

```
subsamp_e_N_01.pl -f C:\sim_data\seriesA\baseR100e0\inds_infiles.txt
-nRi 100 -nRo 40 -e 0.001 -o C:\sim_data\seriesA\sdR40e001
```

The program creates a directory C:\sim_data\seriesA\sdR40e001 and produces outfiles therein. It is important to provide a full path to the outfile directory with flag -o as above. The program produces one fastq file per individual and two additional files: File inds_infiles.txt contains individual IDs and corresponding fastq file names. Provide the path to this file to the program indloc with flag -i when you analyze the simulated dataset with GIbPSs. File subsamp_e_N_rep.txt contains the settings used by the program and a random number seed (rs). You can provide this random number seed with the additional flag -rs to reproduce the same outfiles a second time. We let the program pick a random number seed to create a dataset with 40 read pairs per locus and sequencing error probability 0.001 as above and then used the same random number seed to create additional datasets with 40 read pairs and sequencing error probabilities of 0.002, 0.005, and 0.01. We proceeded accordingly for datasets with 20, 15, and 10 read pairs per locus.

Series B

Create a directory, e.g. C:\sim_data\seriesB and place copies of the Perl scripts sim_dat_07.pl and subsamp_e_N_01.pl in that directory.

Call sim_dat_07.pl as follows:

```
sim_dat_07.pl -nI 100 -nL 1000 -nR 100 -Lsl 120 -Ld 20 -Rl 80 -nM 1
-e 0 -N 0 -i 0 -d 0 -rc 1 -o C:\sim_data\seriesB\baseR100e0
```

The program creates a directory C:\sim_data\seriesB\baseR100e0 and produces outfiles therein. It is important to provide a full path to the outfile directory with flag -o as above. With the above settings, the program simulates a dataset with 100 individuals, 1000 variable loci, and 100 read pairs without sequencing error from each locus and individual. The program simulates alleles of variable loci with one mutation step at a randomly selected position.

Use the Perl script subsamp_e_N_01.pl to subsample the dataset with sequencing error as described under series A. Example:

```
subsamp_e_N_01.pl -f C:\sim_data\seriesB\baseR100e0\inds_infiles.txt
-nRi 100 -nRo 40 -e 0.001 -o C:\sim_data\seriesB\sdR40e001
```

Series C

Create a directory, e.g. C:\sim_data\seriesC and place copies of the Perl scripts sim_dat_07.pl and subsamp_e_N_01.pl in that directory.

Call sim_dat_07.pl as follows:

```
sim_dat_07.pl -nI 100 -nL 1000 -nR 100 -Lsl 120 -Ld 20 -Rl 80 -nM 6
-e 0 -N 0 -i 0 -d 0 -rc 1 -o C:\sim_data\seriesC\baseR100e0
```

The program creates a directory C:\sim_data\seriesC\baseR100e0 and produces outfiles therein. It is important to provide a full path to the outfile directory with flag -o as above. With the above settings, the program simulates a dataset with 100 individuals, 1000 variable loci, and 100 read pairs without sequencing error from each locus and individual. The program simulates alleles of variable loci with six mutation steps at randomly selected positions.

Use the Perl script subsamp_e_N_01.pl to subsample the dataset with sequencing error as described under series A. Example:

```
subsamp_e_N_01.pl -f C:\sim_data\seriesC\baseR100e0\inds_infiles.txt
-nRi 100 -nRo 40 -e 0.001 -o C:\sim_data\seriesC\sdR40e001
```

Series D

Create a directory, e.g. C:\sim_data\seriesD and place copies of the Perl scripts sim_dat_07.pl and subsamp_e_N_01.pl in that directory.

Call sim_dat_07.pl as follows:

```
sim_dat_07.pl -nI 100 -nL 1400 -nR 100 -Lsl 120 -Ld 20 -Rl 80 -nM 2
-e 0 -N 0 -i 200 -nimin 1 -nimax 3 -ilmin 1 -ilmax 2 -d 200 -rc 1 -o
C:\sim_data\seriesD\baseR100e0
```

The program creates a directory C:\sim_data\seriesD\baseR100e0 and produces outfiles therein. It is important to provide a full path to the outfile directory with flag -o as above. With the above settings, the program simulates a dataset with 100 individuals, 1000 "good" variable loci, 200 variable loci with indel variation, and 200 pairs of paralogous loci, each consisting of one variable and one monomorphic locus. It simulates alleles of variable loci with two mutation steps at randomly selected positions. It introduces one to three deletions of length one or two into one allele of a locus with indel variation.

Use the Perl script subsamp_e_N_01.pl to subsample the dataset with sequencing error as described under series A. Example:

```
subsamp_e_N_01.pl -f C:\sim_data\seriesD\baseR100e0\inds_infiles.txt
-nRi 100 -nRo 40 -e 0.001 -o C:\sim_data\seriesD\sdR40e001
```

Creation of simulated datasets, series E

We created four datasets with read depths 12 and 24 and PCR polymerase error rates 0 and 0.001. In the following, we describe how to use script sim_fastq_02.pl to repeat the simulations. The program needs paired fastq infiles with real sequence data as input. It uses quality scores from these files to simulate sequencing errors. We used the first four pairs of fastq infiles from our real 12 sample mouse lemur sequence dataset as input. Call simfastq_02.pl as follows:

```
sim_fastq_02.pl –nI 50 –nL 10000 –nR 24 –Lsl 140 –Ld 20 –Rl 70 –nM 1
–e 0 –fqi /Path/to/directory/with/fastqinfiles –o /Path/to/outfile-
directory –nsf 3000000 –rc 1
```

With these settings, the program simulates 50 individuals and 10,000 loci with one SNP. Loci have a length of 140. The PCR polymerase error rate is 0. The program simulates 24 read pairs per locus and

individual. Use –e 0.001 to simulate a PCR polymerase error rate of 0.001. Use –nR 12 to simulate 24 read pairs per locus and individual. Reads have a length of 91 nucleotides. Forward reads contain a barcode of six nucleotides, a restriction site (GATCC), and the first 80 nucleotides of the locus. Reverse reads contain a restriction site (GATCC) and the first 86 nucleotides of the locus in reverse complementary orientation. The fastq infiles may be plain text files with extension *.fastq or gzip compressed files with extension *fastq.gz. The program produces paired fastq outfiles in the directory specified with flag –o. This directory may not yet exist when you call the program. Fastq outfiles contain a maximum of 3,000,000 sequences per file. Paths provided under flags –fqi and –o must be full paths (e.g. C:\my_simulations\series_E1).

The program produces several additional files. Some of them are needed by subsequent programs. File sim_fastq_rep.txt contains the settings used by the program and a random number seed (rs). You can provide this random number seed with the additional flag -rs to reproduce the same outfiles a second time. Names of the other outfiles are barcodes.txt, inds_barcodes.txt, ori_gt.txt, ori_popall.txt, ori_poploc.txt, popall.fas, and seq_err_rate.txt. Use file barcodes.txt as barcode file for fdm in analyses with GIbPSs. Use file inds_barcodes.txt as barcode file for pyRAD.

Analysis of the simulated data from series A-D

We analyzed each simulated dataset with GIbPSs version 1.0.0 as described in the manuscript and with default settings where not otherwise specified. You can also use GIbPSs version 1.0.1. This version contains only one additional option, which is not used by default. We then used the Perl script comp_simres_06.pl to compare the inferred genotypes to the expected original ones. For each analysis with GIbPSs, we created a main database directory that contained the programs of GIbPSs, outfiles produced by them and a directory user_files with files that contained settings for specific programs of GIbPSs. Please refer to the documentation of GIbPSs for details. As an example, the main database directory for an analysis of dataset sdR40e001 from series C (see above) could be C:\sim_data\seriesC\sdb01. Dataset sdR40e001 is stored at C:\sim_data\seriesC\sdR40e001 and is a subsample of the base dataset stored at C:\sim_data\seriesC\baseR100e0.

Place a copy of comp_simres_06.pl in the main database directory and call it as follows:

```
comp_simres_06.pl -i C:\sim_data\seriesC\baseR100e0 -o
comp_simres_out
```

The program creates a subdirectory comp_simres_out in the main database directory and produces outfiles therein. It reads in files produced by sim_dat_07.pl stored in C:\sim_data\seriesC\baseR100e0 to gather information about the original genotypes. It reads in files produced by the programs of GIbPSs, which are stored in the main database directory to gather information about the currently selected set of inferred genotypes. The program then identifies matching original and inferred loci by exact sequence identity of at least one allele and compares original and inferred genotypes at each locus in each individual. It can happen that the alleles of one original locus match those of several inferred loci or that the alleles of several original loci match those of one inferred locus. The program excludes such loci from the genotype comparison and counts respective genotypes as incorrect. The outfile loc_counts.txt contains counts of loci in the original and inferred dataset in several categories: A found original locus has one matching locus in the inferred dataset. A missing original locus does not appear in the inferred dataset. An excluded original or inferred locus has been excluded as explained above. An erroneous inferred locus matches

no locus in the original dataset. The file further lists loci with and without indel variation and loci with duplicates (paralogs). The file perc_inf_gt.txt contains percentages of genotypes in different categories for each individual and minimum, maximum, and average percentages across individuals. Percentages are based on the total number of loci in the inferred dataset. Categories are as follows:

Identical: The original and inferred genotypes are identical.

Dropout: The original and inferred genotypes differ by allelic dropout.

Mismatch: At least one allele of the inferred genotype is incorrect.

Missing: The locus is missing in an individual.

Excluded_loc: The locus has been genotyped in an individual but has been excluded.

Err_loc: The locus has been genotyped in an individual but does not match an original locus.

Missing_excl_and_err: The locus is missing in an individual and has been excluded or does not match an original locus.

We counted the following categories as incorrect genotypes: *Dropout*, *Mismatch*, *Excluded_loc*, and *Err_loc*. We counted the following categories as missing genotypes: *Missing* and *Missing_excl_and_err*. We then recalculated the percentage of incorrect genotypes based on the number of actually observed loci in each individual as *percent incorrect /* (*percent identical* + *percent incorrect*) * 100.

The file ind_gt_counts.txt contains counts of genotypes for each individual in the same categories.

Analysis of the simulated data from series E with GIbPSs

We analyzed each simulated dataset with GIbPSs version 1.0.1 as described in the manuscript. We then used the Perl script comp_simres_06.pl to compare the inferred genotypes to the expected original ones as described above for series A-D.

Analysis of the simulated data from series E with pyRAD

We truncated the simulated sequence data to a length of 82 and analyzed each dataset with pyRAD as described in the manuscript. Please refer to the documentation of pyRAD for details. We then used the Perl script pR_comp_simres_05.pl to compare the inferred genotypes to the expected original ones. The script analyzes the *.alleles outfile produced by pyRAD in directory outfiles. Say the name of the file is seriesE1.alleles. Place the script pR_comp_simres_05.pl into the pyRAD working directory that contains directory outfiles and call it as follows:

```
perl pR_comp_simres_05.pl –a outfiles/seriesE1.alleles –s
/path/to/outdirectory/of/sim_fastq_02.pl –o /pR_comp_simres_out –r
70.
```

The directory specified with flag –s must contain the additional outfiles produced by sim_fastq_02.pl. The needed files are ori_popall.txt and ori_gt.txt. The program creates a directory pR_comp_simres_out (which may not yet exist) and produces outfiles therein. The program reads in the *.alleles file. It ignores the spacer and surrounding gap-positions and evaluates only the 140 originally simulated positions, which we also evaluated in analyses with GIbPSs. From each sequence, it takes the first 70 and last 70 nucleotides and converts leading and trailing gaps to N. The first

position may be missing in some loci with few individuals. The program then adds N to the beginning of the sequence. The program identifies matching inferred and original loci by comparisons of allele sequences treating N as equal with ACGT. It excludes inferred loci when an inferred locus matches several original ones or when several inferred loci match the same original locus. Loci that do not match any original loci are counted as erroneous. Next, the program matches inferred and original genotypes treating N as equal with ACGT. An inferred genotype matches if allele A matches one original allele and allele B matches the other original allele. A matching genotype is counted as correct when all originally simulated SNPs of the locus (one in our case) are fully determined. Otherwise, it is counted as missing. Some originally simulated SNPs may not occur in the inferred dataset because the locus is unobserved in too many individuals. The program verifies that and evaluates only SNPs that are expected to occur in the inferred dataset. Non-matching genotypes and genotypes of erroneous and excluded loci are counted as incorrect. Unobserved genotypes (missing data points) are counted as missing.

All outfiles except pR_comp_simres_rep.txt are tab-delimited text tables with header line:

File ind_gt_counts.txt contains counts of correct, incorrect, and missing genotypes for each individual.

File loc_counts contains counts of original and inferred loci in different categories.

File nloc_nind.txt lists the number of loci that are observed (not counted missing) in at least n individuals.

File perc_incorr_gt.txt contains the percentage of incorrect genotypes for each individual and the average, minimum, and maximum across individuals. Percentages for a given individual are based on the total number of loci observed (not counted missing) in this individual.

File perc_miss_gt.txt contains the percentage of missing genotypes for each individual and the average, minimum, and maximum across individuals. Percentages for a given individual are based on the total number of loci in the inferred dataset.

File pR_comp_simres_rep.txt contains the runtime and error messages if any.

Analysis of real data with GIbPSs

We analyzed the real dataset with GIbPSs version 1.0.1 as described in the manuscript with default settings where not otherwise specified. Please refer to the documentation of GIbPSs for details. Next, we executed the program export_tables, which is part of GIbPSs. Export_tables produces outfiles in subdirectory export within the main database directory. We used the outfile locdatasel.txt to obtain summary information about the loci identified by GIbPSs. We used the outfile genotypes.txt to perform pairwise comparisons between individual multilocus genotypes. To this aim, we used the Perl script comp_ind_gt_03.pl. Use it as follows to perform pairwise comparisons between genotypes identified by GIbPSs:

Place a copy of comp_ind_gt_03.pl in the main database directory containing your GIbPSs database and call the program:

```
comp_ind_gt_03.pl
```

The program then asks you for a text file that contains the IDs of paired individuals. Each line in the file must contain the IDs of two individuals separated by TAB. The IDs must be those used in the GIbPSs database. You can store it anywhere and provide the path to the file to comp_ind_gt_03.pl. The program then reads in the file export\genotypes.txt out of export_tables and performs pairwise comparisons of genotypes for each pair defined in the text file that you provided. It produces an outfile comp_ind_gt_sum.txt in subdirectory export in the main database directory. The file contains a tab-delimited table that can be opened with a spreadsheet program. The table contains one column with titles followed by one column for each pair of individuals. The first rows contain the following data:

nloc_total: total number of loci in the exported dataset (file export\genotypes.txt)

nloc_ind1: number of loci in the first individual of a pair as in the column header

nloc_ind2: number of loci in the second individual

nloc_both: number of loci shared by both individuals

We used these data to calculate the *pairwise overlap* and *nestedness* for each pair as follows:

pairwise overlap = 100 * *nloc_both* / (*nloc_ind1* + *nloc_ind2* - *nloc_both*)

pairwise nestedness = 100 * *nloc_both* / *nloc_indmin*

where indmin is the individual with fewer loci in the pair.

The next line contains the *pairwise similarity* i.e. the percentage of loci with identical genotypes among all shared loci. The following lines contain percentages of genotypes among the shared loci in different categories, where "dropout" or "drop" stands for allelic dropout, "id" for identical, and "mis" for mismatch. Allelic dropout here means that the two genotypes have different numbers of alleles and that all alleles of the genotype with fewer alleles occur in the genotype with more alleles. Mismatch means any other kind of different genotypes. "Het" and "hom" stand for heterozygous and homozygous. "Hethom" means that individual 1 is heterozygous and individual 2 is homozygous. The last three lines contain percentages of missing genotypes among all loci in the dataset in individual 1, individual 2, and in both individuals of the pair.

Analyses of real data with pyRAD

We analyzed the real dataset with pyRAD 3.0.1 as described in the manuscript. Please refer to the documentation of pyRAD for details. In brief, we demultiplexed the data with pyRAD (step 1). Next, we merged overlapping paired sequences with PEAR 0.9.6. We then continued analyses of merged and unmerged sequences out of PEAR in separate runs with pyRAD (steps 2-7). We used datatype *pairgbs* for analyses of unmerged sequences. In step 2, pyRAD produced a directory `edits` in the working directory, which contained one file with extension `*.edit` for each individual. These files contained filtered and edited sequences. Some of them were trimmed although we had instructed pyRAD not to keep trimmed reads. The trimmed reads then caused a crash of pyRAD in the final step 7. To circumvent the problem, we removed trimmed reads after step 2 with the Perl script remtrim_02.pl.

Usage:

The script must reside in the working directory that contains directory edits. The working directory must contain the barcode file used by pyRAD in step 1. The barcode file must have the name `inds.barcodes`. The file contains one line per individual with the individual ID and the barcode separated by TAB.

Call the script as follows: `perl remtrim_02.pl`

It produces copies of the `*.edit` files in a new directory `editsnotrim`. The new files do not contain trimmed reads. The script produces two additional files: File `editsnotrim/trimrep.txt` lists the number of removed trimmed reads and retained reads for each individual. File `./runtime.txt` contains a note about the runtime used by the script.

Rename the old directory `edits`, then rename directory `editsnotrim` to `edits` and proceed with step 3 to 7 of pyRAD.

In step 7, pyRAD produced outfiles with extension *.alleles, which contained aligned inferred allele sequences of each individual at each locus. Allele sequences inferred from unmerged sequences contained a spacer (nnnn) between the forward and reverse complement reverse sequences. Gaps (-) around the spacer encoded missing sequence positions in some sequences. These positions were missing because pyRAD did not truncate sequences to equal length after removal of the barcodes, which had different lengths. We used the Perl script pRdespac_01.pl to remove the alignment positions containing the spacer and surrounding gaps. To use the script, call it with the name of the *.alleles file as single argument, e.g.:

```
perl pRdespac_01.pl run1.alleles
```

The script then produces an outfile named nospac_run1.alleles, which contains the sequence alignments of the infile without the spacer and surrounding gap positions.

We then analyzed the locus coverage in the inferred dataset and performed pairwise comparisons of individual multilocus genotypes with the aid of the Perl script pRalleles_06.pl. We jointly analyzed the output from the two pyRAD runs with merged and unmerged sequences as one dataset. To use the script, call it as follows:

```
perl pRalleles_06.pl pairs.txt run1.alleles run2.alleles
```

The first argument, "pairs.txt", is the name of a text file that defines pairs of individuals for pairwise comparison. Each line contains two individual IDs separated by TAB. Use this file to define the four pairs of samples from the same individual in our dataset. The next arguments are the names of the *.alleles outfiles of pyRAD. You can also provide only one of them to analyze only one file. In our analyses, the individual IDs differed in the two *alleles outfiles. PEAR appended ".assembled" to individual IDs when merging sequences. The ID of individual "ind01" thus was "ind01" in one file and "ind01.assembled" in the other. PRalleles removes ".assembled" from individual IDs when reading the data. You can thus provide the original IDs in file pairs.txt.

The script then produces six outfiles, which all contain tab-delimited tables:

indloc.txt

This file contains the number of loci observed in each individual.

coverage.txt

This file contains the following columns:

nInd: number of individuals

nloc: number of loci in exactly that number of individuals

nloccum: number of loci in at least that number of individuals

nloccumperc: percentage of loci in at least that number of individuals

indelgtcount.txt

This file contain the number of genotypes with indel variation between the two alleles, which was always 0.

indel_count.txt

This file contains information about indel variation in the inferred dataset:

nlociwithindel: number of loci with indel variation

nlociwithonlyN-indel: number of loci with indel variation where all indel-positions contain only "N" and "-" as characters

perclocwithindel: percentage of loci with indel variation

perclocwithonlyN-indel: percentage of loci with indel variation where all indel-positions contain only "N" and "-" as characters

nindel: total number of indel positions

nN-indel: total number of indel positions that contain only "N" and "-" as characters

Files idpair_comp.txt and nonidpair_comp.txt

These files contain results from pairwise comparisons of individual multilocus genotypes. The first file refers to all pairs defined in file pairs.txt. The second file refers to all pairs NOT defined in that file. The files contain tables with one line for each pair of individuals and three additional lines with average, minimum, and maximum across all pairs. The columns are as follows:

ind1 and ind2: IDs of first and second individual of a pair

nloc1 and nloc2: number of loci in first and second individual

nlocboth: number of loci observed in both individuals, i.e. shared loci

nloctot: number of loci observed in at least one individual of the pair

pwnest: *pairwise nestedness*: percentage of shared loci among the loci observed in the individual with fewer loci observed

pwover: *pairwise overlap*: percentage of shared loci among all loci observed in at least one individual of the pair

nlocid: number of loci with identical genotypes in both individuals

perclocid: *pairwise similarity*: percentage of loci with identical genotypes among all loci observed in both individuals