NGSphy documentation v1.0

http://github.com/merlyescalona/ngsphy

University of Vigo, Spain, http://darwin.uvigo.es

merlyescalona@uvigo.es

1. About NGSphy

NGSphy is a Python open-source tool for the genome-wide simulation of NGS data (read counts or Illumina reads) obtained from thousands of gene families evolving under a common species tree, with multiple haploid and/or diploid individuals per species, where sequencing coverage (depth) heterogeneity can vary among species, individuals and loci, including off-target or uncaptured loci.

2. Citation

If you use NGSphy, please cite:

- Escalona, M, Rocha S and Posada D. NGSphy: phylogenomic simulation of next-generation sequencing data . *Submitted*.

if running ART cite also:

- Huang W, Li L, Myers JR and Marth, GT. (2012) ART: a next-generation sequencing read simulator. *Bioinformatics* 28 (4): 593-594

if using SimPhy cite also:

- Mallo D, De Oliveira Martins L and Posada D. (2016). SimPhy : Phylogenomic Simulation of Gene, Locus, and Species Trees. *Systematic Biology* 65(2): 334-344.

If using single gene tree inputs, cite also:

- Fletcher, W and Yang Z. (2009) INDELible: A flexible simulator of biological sequence evolution. *Molecular Biology and Evolution*. 26 (8): 1879–88.
- Sukumaran, J and Holder MT. (2010). DendroPy: A Python library for phylogenetic computing. *Bioinformatics* 26: 1569-1571.

3. Input/output files

3.1. Input

[Single gene-tree scenario]

- NGSPhy settings file
- INDELible control file
- Newick file with single gene tree
- ancestral sequence file (optional)
- reference allele file (optional)

[Species-tree scenario]

- NGSPhy settings file
- SimPhy output
- reference allele file (optional)

3.2. Output files

- NGS reads:
  - FASTQ
  - ALN
  - BAM
- read counts:
  - VCF
- sequence alignments:
  - FASTA
- coverage variation
  - CSV
- log files
- bash scripts

4. Installation

4.1 Computer requirements

NGSphy has been developed for Linux/MAC environments with Python 2.7.

4.2 NGSphy

To install NGSphy you need to clone its git repository and download the required third-party software (section 4.2):

```
# 1. Clone NGSphy repository
git clone https://github.com/merlyescalona/ngsphy.git
```

```
# 2. Move to ngsphy/dist folder
cd ngsphy/dist
# 3. Extract files and install version XXX:
tar -xzvf dist/ngsphy-XXX.gz
cd ngsphy-XXX
sudo python setup.py install
```

4.3 Third-party software

4.3.1 ART (for Illumina reads generation)

ART is a set of simulation tools to generate synthetic next-generation sequencing reads. You can download it from:

http://www.niehs.nih.gov/research/resources/software/biostatistics/art/
Version ChocolateCherryCake or later.

Following installation instructions from ART, you can download the binaries or compile the source code. If you decide to compile the source code:

```
# 1. Extract files from the compressed tgz
cd /path/to/art-download
tar -xvf artsrcmountrainier20160605linuxtgz.tgz
# 2. Change current directory to the extracted one
cd art_src_MountRainier_Linux/
# 3. Make sure you have all the dependencies installed and generate the
Makefile
./configure
# 4. Run the Makefile
make
```

4.3.2 INDELible (for sequence generation)

INDELible is an application for sequence simulation. You can download it from:

http://abacus.gene.ucl.ac.uk/software/indelible/
Version 1.03.

In order to get INDELible, you will need to register. It is free software, and is distributed under: GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version. For more information go to http://www.gnu.org/licenses/.

Once the software is downloaded:

1. Unpack the archive on Unix-like systems using:

```
# 1. Change directory to the download folder.
cd /path/to/indelible-download
# 2. Extract file from the compressed file.
tar -xvzf INDELibleV1.03.tar.gz
# 3. Change current directory to INDELible's source code folder
cd /path/to/indelible-download/indelible/src/
```

2. Include the following line at to the top of MersenneTwister.h file.

```
#include <unistd.h>
```

3. Compile INDELible using:

```
# 4. Compile the program.
g++ -o indelible indelible.cpp -lm
```

4.3.3. Modified INDELible (NGSphy version)

This is version of INDELible we have modified to allow the use a given ancestral sequence at the root. It can be obtained from cloning its repository:

```
# 1. Clone the repository
git clone https://github.com/merlyescalona/indelible.git
# 2. Change directory to indelible-ngsphy source code folder.
cd indelible-ngsphy/
# 3. Compile
make
```

4.3.4. SimPhy (multiple gene trees evolved under a species tree)

SimPhy can be obtained from cloning its repository and installing its dependencies. Detailed, information on how to install SimPhy here.

```
# 1. Clone the repository
git clone https://github.com/adamallo/SimPhy.git
```

4.4. Adding NGSphy and third-party software to the path

Once all software has been installed, it must be added to the path.

- First you have to add the lines below to the `~/.bashrc file` to keep the changes permanently.

```
ARTpath="/path/to/art/executable"
INDELIBLEpath="/path/to/indelible/executable"
INDELIBLENPpath="/path/to/indelible-ngsphy/executable"
NGSPHYpath="/path/to/ngsphy/executable"
export PATH="$ARTpath:$INDELIBLEpath:$INDELIBLENPpath:$NGSPHYpath:$PATH"
```

- Apply changes

```
source ~/.bashrc
```

5. Usage

NGSphy does not have a Graphical User Interface (GUI) and works on the Linux/Mac command line in a non-interactive fashion.

```
usage: ngsphy  [-s <settings_file_path>]
               [-l <log_level>] [-v] [-h]
```

- Optional arguments:
  - -s <settings_file_path>, --settings <settings_file_path>
    Path to the settings file
  - -l <log_level>, --log <log_level>
    Specified hierarchical log levels that will be shown through the standard output. A detailed log will be stored in a separate file. Possible values:
    - **DEBUG**: shows very detailed information of the program's process.
    - **INFO** (default): shows only information about the state of the program.
    - **WARNING**: shows only system warnings.
    - **ERROR**: shows only execution errors.

- Information arguments:
  - -v, --version
    Show program's version number and exit.
  - -h, --help
    Show help message and exit.

Some simple examples:

1. When there is `settings.txt` file in the current working directory.

```
ngsphy
```

2. Run with an specific settings file `my_settings.txt`

```
ngsphy -s my_settings.txt
```

6. The settings file

NGSphy requires a settings file "**settings.txt"** that specifies the different options and parameter values for the simulations. A settings file with a different name can be specified with the **-s/--settings** option. The information in the settings file is organized in 6 optional/required blocks (default values are underlined):

1. **[general]**: general parameters.
2. **[data]**: specifies the type of input data as well as input parameters and files.
3. **[coverage]**: parameters that describe the variation of coverage in the dataset (optional).
4. **[ngs-reads-art]**: specifies ART execution parameters (optional)
5. **[ngs-read-counts]**: specifies parameters for read counts (optional).
6. **[execution]**: describes how the execution of the whole process will be made (optional).

6.1. [general] block

Stores general parameters for each NGSphy run.

```
[general]
path=/home/user/
output_folder_name=NGSphy_output
ploidy=1
```

- **path**
  - purpose: path where output folder will be created.
  - type: string (path)
- **output_folder_name**
  - purpose: name of the output folder where NGSphy results will be stored. If the output folder already exists, the new output folder will get the same base name

with a numerical suffix (outputFolder_n), representing the nth time the program with that output folder name was ran.
    - type: string
    - value: NGSphy_output
- **ploidy**
    - purpose: refers to the ploidy that the resulting individuals will have. So far it is only possible to generate haploid and diploid individuals.
    - type: number (integer)
    - values: 1, 2 (in the closed-interval [1,2])

6.2. [data] block

Defines the input data for NGSphy, which consists of 3 different modes:

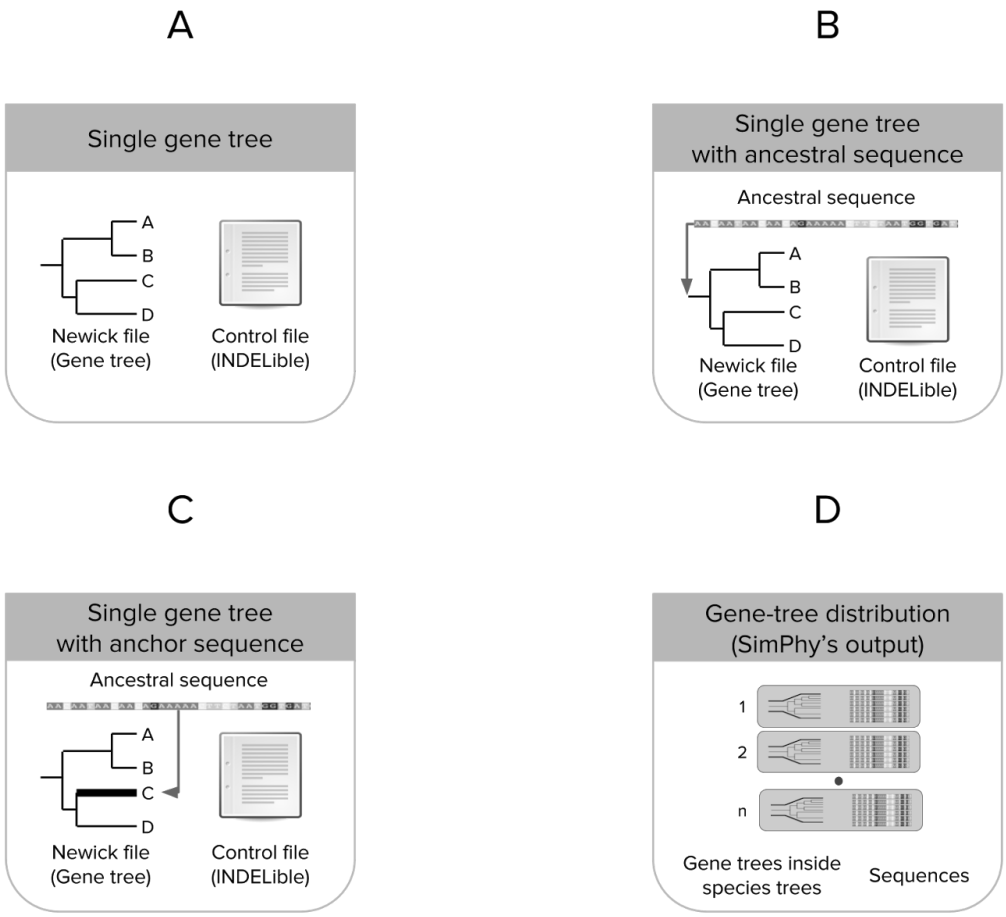


FIGURE 1: Input modes: a) a single gene tree; b) single gene tree with a user-defined ancestral sequence; c) a single gene tree with an anchor sequence and d) gene-tree distributions (SimPhy output [species-tree simulations])

6.2.1. Input data options

Single gene tree	inputmode=1 gene_tree_file=/home/myuser/my_gene_tree.tree indelible_control_file=/home/myuser/my_control_indelible.txt
Single gene tree with user-defined ancestral sequence	inputmode=2 gene_tree_file=/home/myuser/my_gene_tree.tree ancestral_sequence_file=/home/myuser/my_ancestral.fasta indelible_control_file=/home/myuser/my_control_indelible.txt
Single gene tree with user-defined anchor sequence	inputmode=3 gene_tree_file=/home/myuser/my_gene_tree.tree anchor_sequence_file=/home/myuser/my_anchor.fasta anchor_tip_label=1_0_0 indelible_control_file=/home/myuser/my_control_indelible.txt
Gene-tree distribution SimPhy output (species-tree simulations)	inputmode=4 simphy_folder_path=testSimphy simphy_data_prefix=data simphy_filter=true

- **inputmode**
  - purpose: identifies the type of input.
  - type: number (integer)
  - value: values within the closed interval [1,3]
    1. single gene tree
    2. single gene tree with an ancestral sequence
    3. single gene tree with an anchor sequence
    4. gene-tree distribution (SimPhy output [species-tree simulations])

6.2.1. Single gene tree

- **gene_tree_file**
  - purpose: path of the gene tree in Newick format . There must be a single path and a single tree in the file. The name of the file, without extension, must be the same as the name of the tree within the INDELible control file, in the **[NGSPHYPARTITION]** option.
  - type: string (path)
  - format: see specification in Section 6.2.4. (INDELible control file).
- **indelible_control_file**
  - purpose: path for the INDELible control file.
  - type: string (path)
  - format: see specification in Section 6.2.4. (INDELible control file).

6.2.2. Single gene tree with an user-defined ancestral sequence

These options are related to the INDELible run with a user-defined ancestral sequence:

- **gene_tree_file**
  - purpose: same as in Section 6.2.2 (Single gene tree)
  - type: string (path)
- **ancestral_sequence_file**
  - purpose: path to the FASTA file that contains the ancestral sequence.
  - type: string (path)
- **indelible_control_file**
  - purpose: Same as Section 6.2.2.
  - type: string (path)

6.2.3. Single gene tree with an user-defined anchor sequence

These options are related to the INDELible run with a user-defined ancestral sequence:

- **gene_tree_file**
  - purpose: same as in Section 6.2.2 (Single gene tree)
  - type: string (path)
- **anchor_sequence_file**
  - purpose: path to the FASTA file that contains the anchor sequence.
  - type: string (path)
- **anchor_tip_label**
  - purpose: tip label of the gene tree that corresponds to the tip that will be used as root.
  - type: string
  - format: see specification in the Section 6.2.5 (Single gene-tree file labeling)
- **indelible_control_file**
  - purpose: Same as Section 6.2.2.
  - type: string (path)

6.2.4. Gene-tree distribution (SimPhy output [species-tree simulations])

- **simphy_folder_path**
  - purpose: path to the folder with SimPhy's output
  - type: string (path)
- **simphy_data_prefix**
  - purpose: prefix used in SimPhy's run.
  - type: string
- **simphy_filter [optional]**
  - purpose: filter out the replicates that do not satisfy the required ploidy. For the diploid case the number of gene tree tips per species has to be an even number. See more in Section 6.2.7. (Individual generation).
  - type: boolean
  - value:
    - **0, false, off:** don't filter
    - **1, true, on**: filter

6.2.4.1. A valid SimPhy ouput

A detailed description of SimPhy's output can be found https://github.com/adamallo/simphy. The SimPhy output required by NGSphy has to include:

- **<simphy_project_name>.command:** a plain text file with the original command line arguments.
- **<simphy_project_name>.db:** a SQLite database composed by three (3) linked tables with different information about species, locus and gene trees.
- **<simphy_project_name>.params:** a plain text file summarizing the sampled options.
- a set of folders with the multiple sequence alignments and the corresponding trees.

6.2.5. INDELible Control file - NGSphy version

When the input mode is a single gene tree, it is necessary to have a control file to call INDELible. Here, we use a slightly modified version of the INDELible's control file. To properly set up the configuration file for INDELible, users should refer first to INDELible's manual (http://abacus.gene.ucl.ac.uk/software/indelible/manual/)l. In our version, the file must include the following blocks:

- **[TYPE]**: 1 block
- **[SETTINGS]**: 1 block (optional)
- **[MODEL]**: 1 block
- **[NGSPHYPARTITION]**: 1 block
- **[NGSPHYEVOLVE]**: 1 block

Including a wrong number of blocks or other type of blocks will result in an error message and will terminate NGSPhy execution.

6.2.4.1. Block definitions

- **[TYPE]** standard INDELible specification.
- **[SETTINGS]** standard INDELible specification.
- **[MODEL]** standard INDELible specification.
- **[NGSPHYPARTITION]** this block defines:
    - the gene tree for INDELible (this name has to be the same as the Newick file used as input (see Section 6.2)
    - the substitution model for INDELible. This name must match the name of the model used in the previous **[MODEL]** block.
    - the sequence length.

  For example, we have a gene tree in the Newick file: **tree1.tree**, where sequences will evolve under model **m1**, with a length of 500bp.

```
[NGSPHYPARTITION]  tree1 m1 500
```

- **[NGSPHYEVOLVE]** block:

- the prefix for the output filenames

For example, one sequence alignment for each gene tree, saved in files with 'dataset' as common prefix. This will generate the following filenames: dataset_1, dataset_2...

```
[NGSPHYEVOLVE]  dataset
```

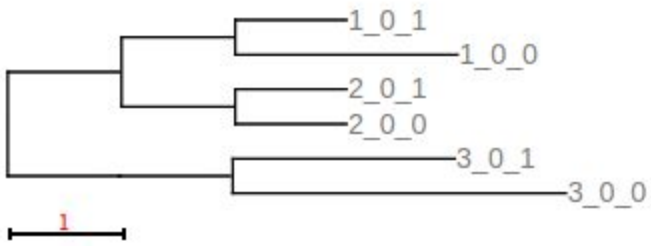
6.2.5. Single gene-tree file format and labeling

Single gene trees in Newick format should have specific tip labels. Tips must follow a specific format in order to be managed by NGSphy. This format indicates species, locus and individual with the scheme (**X_Y_Z**) where:

- **X** stands for the the species identifier, where **X > 1**
- **Y** for the locus identifier, where **Y > 0**
- **Z** for the individual identifier, where **Z> 0**

The gene tree file must be in Newick format, rooted and with branch lengths. If the gene tree is not rooted, it will be forced following Dendropy specifications.

For example, if we have 3 species and 2 gene copies per species the labels would be:




```
(((1_0_1:1.0,1_0_0:2.0):1.0, (2_0_1:1.0,2_0_0:1.0)),( (3_0_1:2.0,3_0_0:3.0) ));
```

6.2.7. Individual assignment

For haploid individuals, each tip in the gene tree provided will correspond to a single individual. For diploid individuals the number of gene-tree tips per species must be even. In this case, the individuals are generated by randomly sampling without replacement two gene copies from a specific gene-family until all gene tree tips have been assigned to an individual.

For the gene-tree distribution input mode only, the outgroup in the gene trees is called "**0_0_0"** and has one gene copy. Therefore, for the generation of diploid individuals, the outgroup will be homozygous, obtained by the duplication of the sequence of its gene copy.

6.3. [coverage] block

Sequencing coverage can be specified at three different levels: experiment, individual and locus-wide. It is also possible to mimic the variation in coverage expected for targeted sequencing, including off-target loci and taxon-specific effects.

```
[coverage]
experiment=F:100
individual=LN:1.2,1
locus=LN:1.3,1
offtarget=0.4, 0.01 # 40% loci are off-target, will have 1% of the coverage
notcaptured=0.5
taxon= 1,0.5;2:0.25
```

6.3.1. Sampling notation

The parameters that will define the coverage in NGSphy have to be provided using a specific notation in order to define statistical distributions and dependency between arguments. The sampling notation is structured as a particular statistical distribution (see code for the statistical distribution), followed by a colon and a list of comma-separated parameter values:

```
distribution_code:param1,param2, ...
```

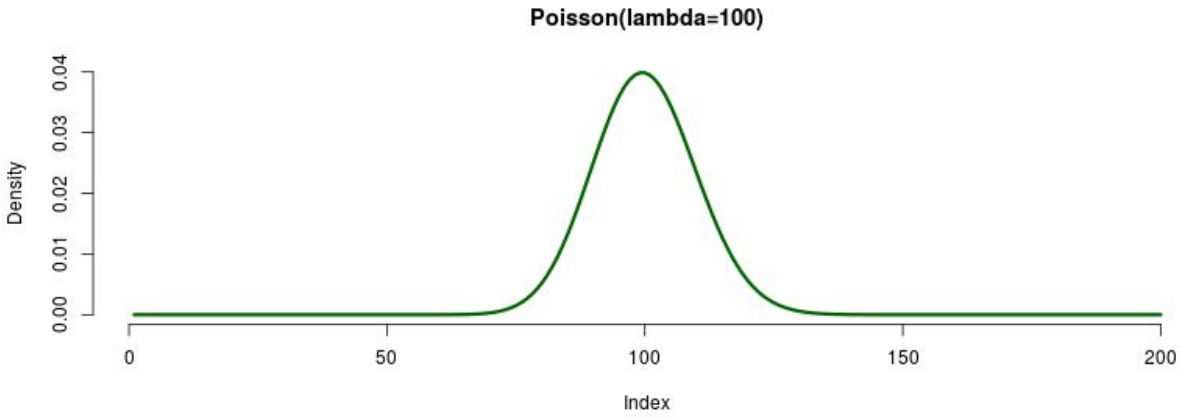
For example:

a) Fixed value=100.

```
F:100
```

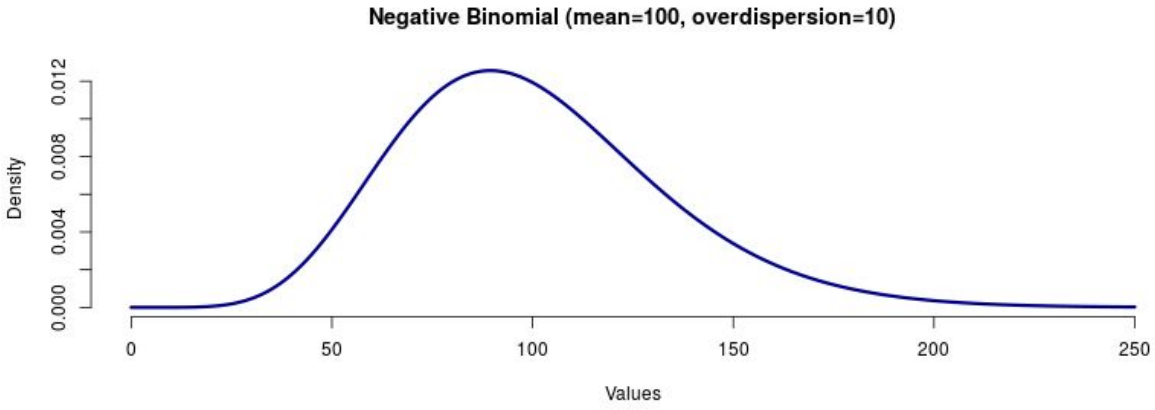
b) Poisson distribution with mean=100.

```
P:100
```



c) Negative Binomial, mean=100 and overdispersion=10.

```
NB:100,20
```



6.3.2. Statistical distributions

Distribution	Code	Num. parameters	Parameters	Description
Binomial	b/B	2	r,p	trials, probabilities
Exponential	e/E	1	s	scale
Fixed point	f/F	1	v	value
Gamma	g/G	2	sh,sc	shape,scale
Normal	n/N	2	mu, var	mean, variance
Log. Normal	ln/LN	2	mu, sd	mean, standard deviation
Negative Binomial	nb/NB	2	mu, r	mean of the underlying Poisson distribution, overdispersion
Poisson	p/P	1	l	mean
Uniform	u/U	1	mu	mean

6.3.3. Coverage options

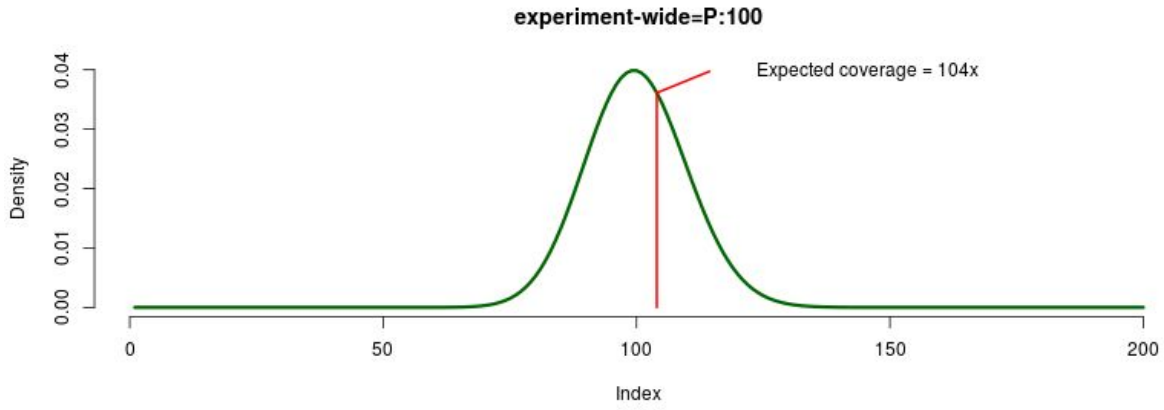
- **experiment**
  - purpose: expected depth of coverage for a specific replicate.
  - type: fixed value or statistical distribution.
- **locus [optional]**
  - purpose: variation of expected coverage between loci.
  - type: fixed value or statistical distribution**.**
- **individual [optional]**
  - purpose: variation of expected coverage between individuals.
  - type: fixed value or statistical distribution**.**
- **offtarget [optional]**
  - purpose: related to targeted-sequencing experiments; percentage of loci that will be considered off-target (captured and sequenced but not originally targeted); expected coverage will be 1% of the experiment-wide.
  - type: 1 pair (proportionLoci, proportionCoverage)
  - value:
    - proportionLoci: number (float) in the closed interval [0,1].
    - proportionCoverage: number (float) in the closed interval [0,1].
- **notcaptured [optional]**
  - purpose: related to targeted-sequencing experiments; fraction of originally targeted loci that will not be captured/sequenced.
  - type: number (float).
  - value: number in the closed interval [0,1].

- **taxon [optional]**
    - purpose: related to targeted-sequencing experiments; decrease in coverage for particular species. It can be due to the phylogenetic distance between a reference species (used to design the probes for the targeted loci) and the individuals from the target-sequencing experiment or to species-specific sample conditions.
    - type: pairs (speciesID,coverageProportion)
    - values:
        - **speciesID**: one or more of the existent species in the tree.
        - **coverageProportion**: value in the closed interval [0,1].
    - format:

```
taxon=speciesID1,coverageProportion1; speciesID2,coverageProportion2 ...
```

6.3.4. Coverage sampling strategy

The **experiment-wide coverage** is sampled for each replicate from the specified statistical distribution, and this value becomes the expected coverage for every loci and individual in that replicate. For example, if experiment-wide=P:100, we might sample a value of 104 for replicate 1, so the expected coverage would be 104x for that particular experiment.



An **individual-wide** coverage multiplier is sampled for each individual within a given replicate. The value indicated in the settings file is in fact a hyper-parameter that controls a specific hyper-distribution from which a single value is sampled per replicate. For that replicate, this value will become the shape of a Gamma distribution with mean = 1, from which a multiplier is sampled for each individual.

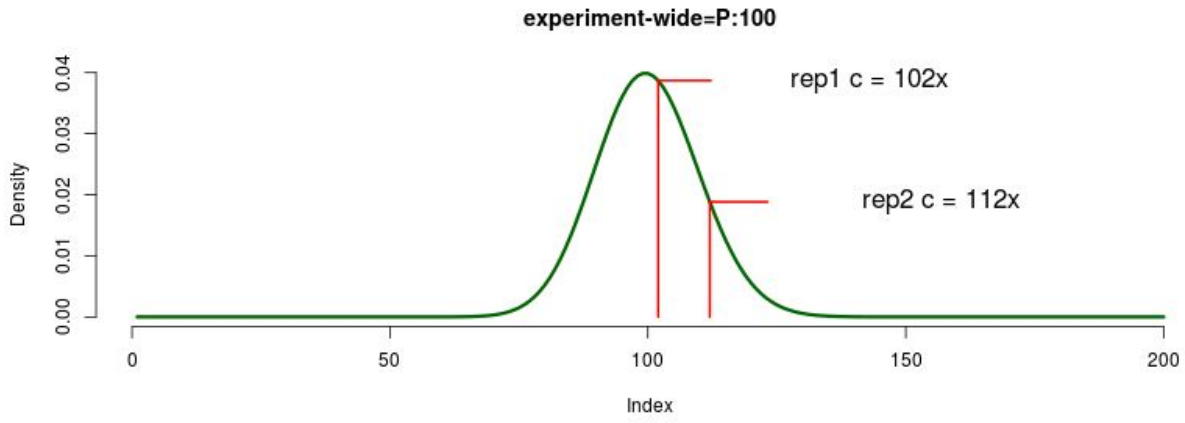
In exactly the same manner, a **locus-wide** coverage multiplier is sampled for each locus within a given replicate. The value indicated in the settings file is again a hyper-parameter that controls a specific hyper-distribution from which a single value is sampled per replicate. For that replicate, this value will become the shape of a Gamma distribution with mean = 1, from which a multiplier is sampled for each loci.

For example, imagine we have 2 replicates, 2 loci, 2 individuals and input the following coverage settings:

```
[coverage]
experiment-wide: P:100
locus-wide: LN:1.2,1
individual-wide: E:1
```

First, we sample from a Poisson, with mean=100, to obtain the expected coverage per experiment (rep1c, rep2c).

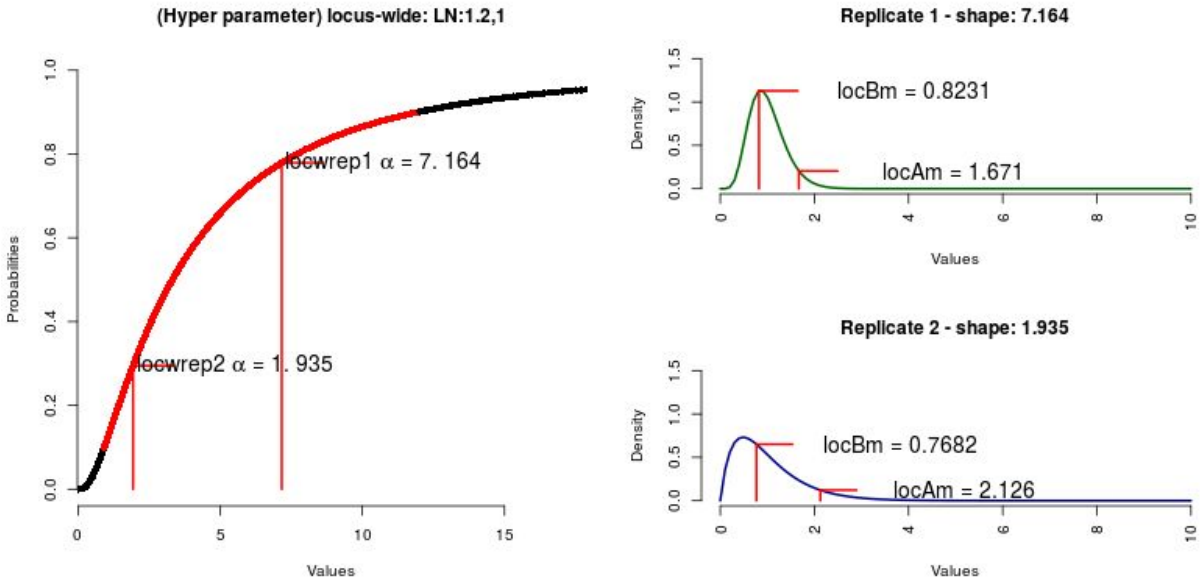
I



- Coverage variation before locus/individual multipliers:

Replicate	Locus	Expected coverage	
		Individual I	Individual II
1	A	102	102
	B	102	102
2	A	112	112
	B	112	112

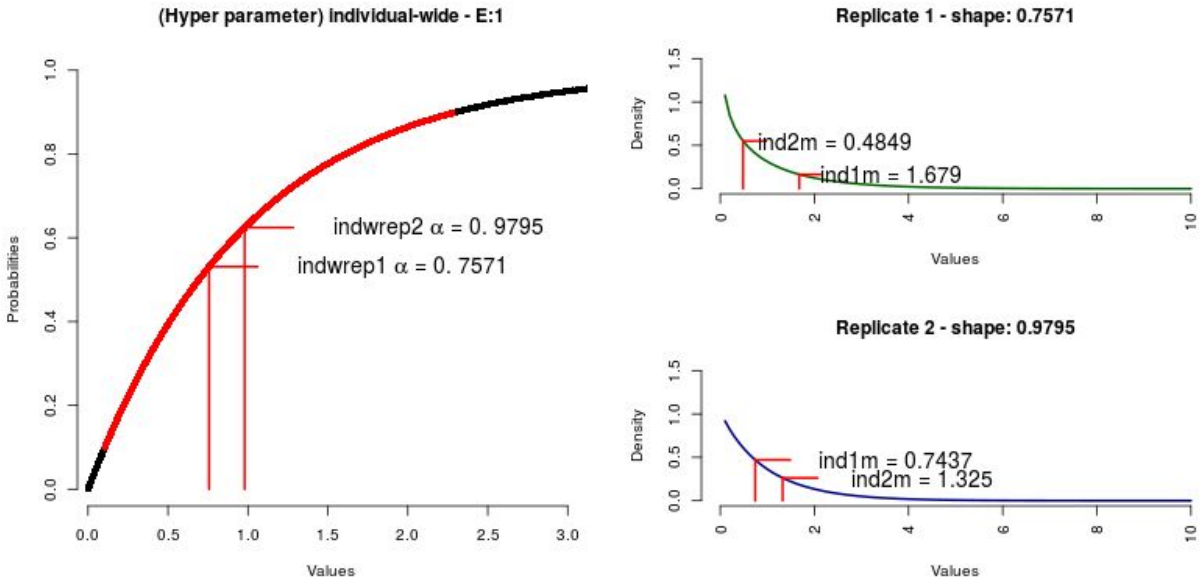
- Afterwards, we sample the **locus-wide** rate multipliers from the hyper-distribution, in this case a Log Normal with mean=1.2 and standard deviation =1 ($locrep1_{\alpha}$,$locrep2_{\alpha}$). This, give us the shape of the Gamma distribution with mean 1 from which we sample the rate multipliers, as many as loci (locAm, locBm).



- Coverage variation after locus-wide multipliers:

Replicate	Locus	Rate multiplier (per loci)	Resulting coverage	
			Individual I	Individual II
1	A	1.671	170.4420	170.4420
	B	0.8231	83.9562	83.9562
2	A	2.1260	238.1120	238.1120
	B	0.7682	86.0384	86.0384

- Next, we get the **individual-wide** rate multipliers, sampling from the hyper-distribution, an Exponential with rate 1 (indwrep1 α, indwrep2 α). This, give us the shape of the Gamma distribution with mean 1 from which we sample the rate multipliers, as many as individuals (indAm, indBm).



Finally, we apply all the multipliers. Coverage variation after locus-wide and individual-wide multipliers:

Replicate	Individuals	Rate multiplier (per individual)	Resulting coverage	
			locus A	locus B
1	I	0.4849	82.64733	40.71036
	II	1.679	286.1721	140.9625
2	I	0.7437	177.08389	63.98676
	II	1.325	315.4984	114.0009

Targeted sequencing parameters allow the user to emulate the variation in depth of coverage that can occur in a targeted-sequencing experiment. This is possible when using gene tree distributions (SimPhy project) as input data. These parameters identify the on-target/off-target loci as well as the number of loci that may not be captured. While on-target loci will keep their expected coverage, the off-target fraction will have a (user-defined) fraction of this. The not-captured indicates the fraction of targeted loci that will not be captured, and its expected coverage will be 0x. For example, if we have 2 replicates, 3 loci, and input the following coverage:

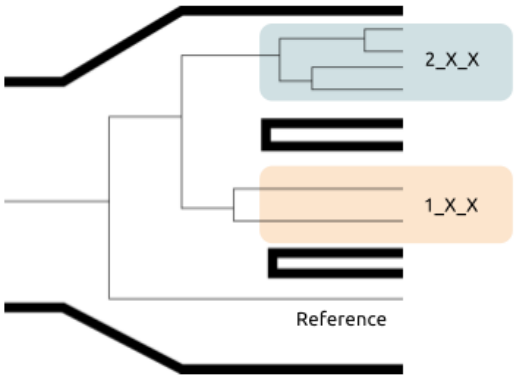
```
[coverage]
experiment-wide: P:100
```

```
off-target=0.33 0.1
notcaptured=0.5 # half of the on-target
```

If we consider the same coverage sampling as before, P:100:

Replicate	Locus	Category	Expected coverage	Rate multiplier	Sampled coverage
1	A	on-target	105	1	105
	B	on-target, not captured	105	0	0
	C	off-target	105	0.1	10.5
2	A	on-target	92	1	92
	B	on-target, not captured	92	0	0
	C	off-target	92	0.1	9.2

Taxon-specific effects allows the user to define of coverage variation for specific taxa. It can be used for example to emulate a decay in coverage, related to the phylogenetic distance of the a species to the reference species used to build the target-loci probes (Bragg et al, 2016) (this is sometimes called phylogenetic decay) or in a more general context for particular sample conditions (low amount of DNA, museum specimens, etc.).



For example:

```
[coverage]
experiment: F:60
taxon=1,0.5; 2,0.25
```

Meaning that, if the expected coverage for the experiment is 60x, individuals from the species speciesID=1, will have a coverage of 30x (50% of the expected coverage) and the individuals from the species speciesID=2, will have coverage of 15x (25% of the expected coverage).

6.4. [ngs-reads-art] block

Defines the options for ART. If the user specifies here any input (in,i), output (out,o) or coverage related options (fcov, f, rcount, c), these will be ignored.

6.5. [ngs-read-counts] block

Generates a VCF file per locus per replicate, that contains the variable positions, haplotype/genotype and likelihoods.

```
[ngs-read-counts]
read_counts_error
reference_sequences_file
```

- **read_counts_error**
  - purpose: to emulate sequencing error.
  - type: number (float)
  - value: value in the left-closed interval [0,1).
- **reference_alleles_file**
  - purpose: identifiers of the sequences used as reference for the variable sites.
  - type: string (path)

6.5.1. Reference allele file [optional]

Defines which alleles will be used as references to generate the VCF files. The description of the allele sequences follow the labeling explained above in Section 6.2.5 (Single gene-tree file labeling). The content of the file should be formatted as:

```
REPID, SPID,LOCID, INDID
```

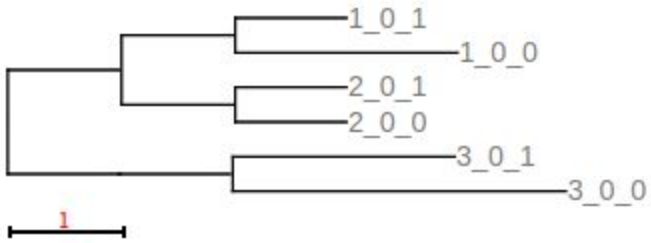
Where:

- **REPID**, replicate ID.
- **SPID**, species ID (X value of the sequence description)
- **LOCID**, locus ID (Y value of the sequence description)
- **INDID**, gene tree tip ID (Z value of the sequence description).

IMPORTANT: By default, if the reference allele file is not specified (or badly formatted), the reference allele will correspond to the sequence named 1_0_0.

6.5.1.1. Example

The simplest case will be when the input is a single tree and all individuals have the same number of loci. So, let's suppose we want to run NGSphy, with single gene tree inputmode (inputmode=1). The gene tree is the following (as in Section 6.2.5) :



Also, we want to generate read counts, with no errors, and we want to use the gene-tree tip with label "**2_0_1"** as the reference allele. And so, the reference allele file should contain:

```
1,2,0,1
```

6.6. [execution] block

This section define how NGSphy is executed. If the user has access to a computational cluster, the ART commands can be converted into jobs for SGE or SLURM schedulers (see Section 6.6.2). If desired, ART calls can be made by NGSphy transparently to the user (sequentially or in parallel - multi-threading).

```
[execution]
environment=bash
runART=on
running_times=off
threads=4
```

6.6.1. Options

- **environment**
  - purpose: specify in which environment the ART runs are going to be executed (more details below)
  - type: enumerate (possible environments)
  - values:
    - **bash:** generates a bash file with all the commands used to call ART.
    - **sge:** generates the necessary files to run a job array in a cluster environment running Sun Grid Engine. Includes: seed file, job script and a possible script to launch ART jobs.

        - **slurm:** generates the necessary files to run a job array in a cluster environment running Simple Linux Utility for Resource Management. Includes: seed file, job script and a possible script to launch ART jobs.

- **threads**
    - purpose: number of threads to execute NGSphy.
    - type: number (integer)
    - value: 1
- **runART**
    - purpose: indicate whether the user actually wants to generate NGS reads This will only run on local, under bash environment
    - type: boolean
    - values:
        - **1, true, on**: run ART.
        - **0, false, off:** don't run ART, bash scripts will be generated.
- **runnintg_times**:
    - purpose: obtain the running times file for the NGS mode processes (read counts or ART).
    - type: boolean
        - values:
            - **0, false, off:** don't generate file
            - **1, true, on**: generate file

IMPORTANT: the generation of this file increases the execution time of the program.

    -

NOTES

- If the execution block is missing, a bash script will be generated and ART instances will not be run.
- If the option environment is missing, a bash script will be generated (default behavior) and ART instances will not be run, unless run option is set.
- If the option run is missing, ART instances will not be run.
- If the value chosen for the option run is wrong and bash is the value of environment, then ART instances will not be run.
- If the value chosen for the option environment is wrong, behavior will be as if there was no execution section, bash script will be generated and ART instances will not be run.

6.6.2. Cluster execution options (SGE,SLURM)

NGSphy can generate job templates for execution in computational clusters running Sun Grid Engine (Gentzsch 2001, Oracle Corp.) or Simple Linux Utility for Resource Management (Yoo et al. 2003, https://slurm.schedmd.com/).

In this case, NGSphy generates two files, XXX (job script) and YYY (seed-file for job arrays). To execute this one would type a different command depending on job scheduler (SGE or SLURM)

- SGE:

```
qsub -t 1-100 project.sge.sh
```

- SLURM:

```
sbatch --array 1-100 project.slurm.sh
```

Here there are some arbitrary examples of the files generated:

- SEED FILE

```
<input_file>    <output_file>
```

- SGE job script:

```
#!/bin/bash
# SGE submission options
#$ -l num_proc=1          # number of processors to use
#$ -l h_rt=00:10:00       # Set 10 mins  - Average amount of time for up to 1000bp
#$ -t 1-{0}               # Number of jobs/files that will be treated
#$ -N art.sims            # A name for the job

inputfile=$(awk 'NR==$SGE_TASK_ID{{print $1}}' $SEEDFILE)
outputfile=$(awk 'NR==$SGE_TASK_ID{{print $2}}' $SEEDFILE)\n

art_illumina -ss GA2 -amp -p -sam -na -i $inputfile -l 50 -f 10 -o $outputfile
```

- SLURM job script

```
#!/bin/sh
#SBATCH -n 1
#SBATCH --cpus-per-task 1
#SBATCH -t 00:10:00
#SBATCH --mem 4G
#SBATCH --array=1-1000

inputfile=$(awk 'NR==$SLURM_ARRAY_TASK_ID{{print $1}}' $SEEDFILE)
outputfile=$(awk 'NR==$SLURM_ARRAY_TASK_ID{{print $2}}' $SEEDFILE)
```

```
art_illumina -ss GA2 -amp -p -sam -na -i $inputfile -l 50 -f 10 -o $outputfile
```

IMPORTANT: Take into account that the job script files generated by NGSphy are general templates, and that in most cases they will be have to be modified according the the particular cluster environments. It is strongly encouraged to consult the cluster administrator for proper execution.

6.6.3. Running times file

Generated to keep track of the timings for each ART call or each NGS read counts process. File name follows the format:

```
project.info
```

where, **project** will be **NGSphy,** if using any of the single gene tree input modes. Whereas, for the gene-tree distribution input mode, it will be the name of the SimPhy output folder.

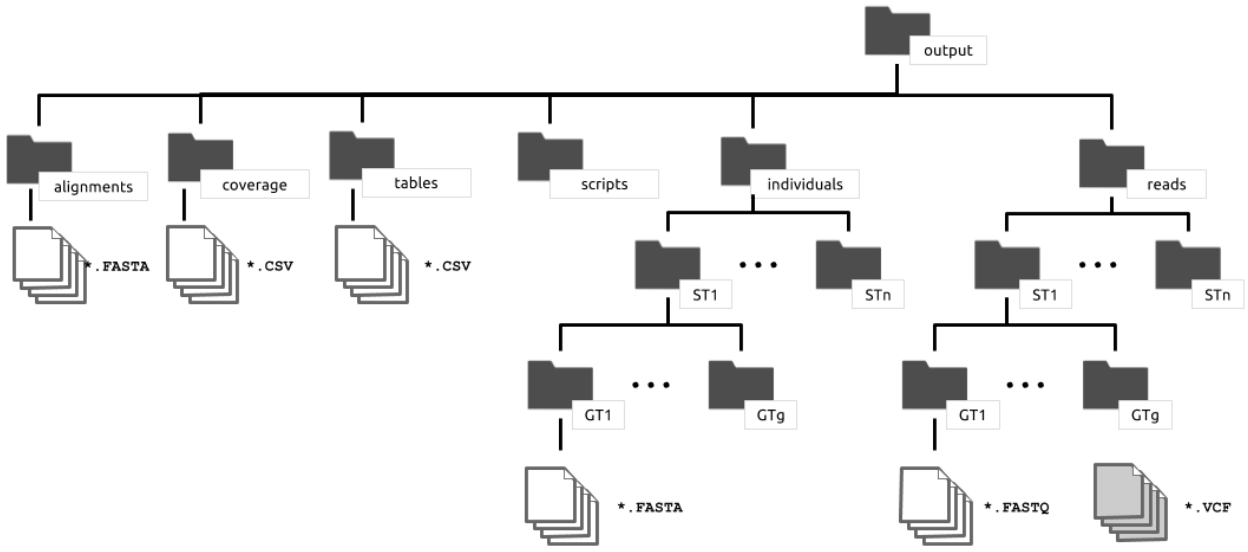
Content of the file is formatted as follows:

```
indexREP,indexLOC,indID,inputFile,cpuTime,seed,outputFilePrefix
```

- **indexREP**: replicate identifier.
- **speciesID**: species identifier.
- **locusID**: locus identifier.
- **indID:** individual identifier.
- **inputFile**: path of the input file, corresponding to the individual FASTA file.
- **cpuTime**: processing time
- **seed**: if the NGS mode needs a seed for the generation of random numbers, it will be here.
- **outputFilePrefix**: prefix of the file generated.

7. Output

The output of NGSphy will depend on the NGS mode selected (read-count or ngs-reads-art). In both cases, the user will get a detailed log file and a folder structure as:



Folder structure include:

1. **alignments**: for single gene tree modes, stores the alignments and files generated for the INDELible run.
2. **coverage**: stores tables describing the coverage for each locus and individual, one per replicate.
3. **individuals**: stores the FASTA files with the individual sequences. Structured along the hierarchy replicate > locus > individuals.
4. **ind_labels**: stores the correspondence between sequences and individuals.
5. **reads**: for Illumina reads, stores the ALN/BAM and/or FASTQ files generated by ART. For read counts, stores all the VCF files. Structured as hierarchy:
    a. replicate > locus > ALN/BAM/FASTQ/VCF files
6. **ref_alleles**: stores the sequences of the references alleles used for the simulation of read counts.
7. **scripts**: stores all the bash scripts generated.

7.1. Alignments

Stores the simulated alignments in FASTA format (if indels are simulated, there will be a TRUE file with the true alignment), together with INDELible/ INDELible-ngsphy control file, ancestral sequence, and gene trees.

```
alignments/
    |__ngsphy.tree # if inputmode = 3
    |__NGSphy.indelible.times # if running_times=1
    |__1
        |__control.txt
        |__ancestral.fasta # if inputmode in [2,3]
```

```
        |__ngsphydata_1.fasta
        |__ngsphydata_1_TRUE.fasta
        |__LOG.txt # default indelible file
        |__tree.txt # default indelible file
```

NOTE: During the simulation process, the way it is implemented, the anchor sequence in the alignment produced by INDELible might include indels.

7.2. Coverage

This folder will contain comma-separated file (CSV) files with the coverage distribution of each individual per replicate. Each file stores a matrix of shape (number of individuals X number of loci) where each cell corresponds to the depth of coverage of the loci for the specific individual. Format of the filename is:

```
project.repID.csv
```

Where:

- **project**: if using any of the single gene tree input modes, it will be **NGSphy**. For the gene-tree distribution input mode, it will be the name of the SimPhy output folder.
- **repID**: number of the replicate.

Folder structure will look like this:

```
coverage/
    |__SimOhyOutput.1.csv
    |__SimOhyOutput.2.csv
    |__SimOhyOutput.3.csv
    ...
```

7.3. Ind_labels

These will store the correspondence between the original sequences and the generated individuals. Each table is a CSV file named as follows:

```
project.repID.individuals.csv
```

Where:

- **project**: if using any of the single gene tree input modes, it will be **NGSphy**. For the gene-tree distribution input mode, it will be the name of the SimPhy output folder.
- **repID**: number of the replicate.

Folder structure will look like this:

```
ind_labels/
    |_NGSphy.1.individuals.csv
```

7.3.1. Haploid individuals

This folder will contain tables with the correspondence between the individual identifier and the corresponding sequence identifier. CSV file format:

```
REPID, indID, speciesID,locusID,geneID
1,  0,  0,  0,  0
1,  1,  1,  0,  1
1,  2,  1,  0,  2
```

Where:

- **REPID**: identifier of the replicate to which the gene trees and sequences belong.
- **indID**: identifier of the haploid individual.
- **speciesID**: identifier of the species
- **locusID**: identifier of the locus
- **geneID**: identifier of the gene tree tip.

7.3.2. Diploid individuals

These tables will contain the correspondence between each individual and its two sequences. CSV file format:

```
REPID,indID,speciesID,locusID,mateID1,mateID2
1,  1,  1, 0, 3,  0
1,  2,  1, 0, 4,  1
1,  3,  1, 0, 2,  5
1,  4,  3, 0, 4,  0
```

Where:

- **REPID**: identifier of the replicate.
- **indID**: identifier of the generated diploid individual.
- **speciesID**: identifier of the species.
- **locusID**: identifier of the locus.
- **mateID(1&2):** identifier of the gene tree tip used for the 1(2) sequence of the individual.

7.4. Individuals

This folder will store the diploid individual sequence files (i.e., 2 sequences for each locus), hierarchically organized within replicates and loci. For example:

```
individuals/
```

```
  |__1/
    |__1/
      |__prefix_1_1_ind1.fasta
      |__prefix_1_1_ind2.fasta
      |__prefix_1_1_ind3.fasta
    |__2/
      |__prefix_1_2_ind1.fasta
      |__prefix_1_2_ind2.fasta
      |__prefix_1_2_ind3.fasta
  |__2/
    |__1/
      |__prefix_2_1_ind1.fasta
      |__prefix_2_1_ind2.fasta
      |__prefix_2_1_ind3.fasta
    |__2/
      |__prefix_2_2_ind1.fasta
      |__prefix_2_2_ind2.fasta
      |__prefix_2_2_ind3.fasta
```

7.5. Ref_alelles

This folder contains the FASTA files with the reference allele sequences used in the VCF file with the read counts. Folder is structured per replicate. There is a reference allele file per locus. Each file contains a single sequence. The format of each file name:

```
project_REF_repID_locID.fasta
```

Where:

- **project**: if using any of the single gene tree input modes, it will be **NGSphy**. For the gene-tree distribution input mode, it will be the name of the SimPhy output folder.
- **repID**: replicate identifier.
- **locID**: locus identifier.

Folder structure will look like this:

```
ref_alleles/
  |__1/
    |__NGSphy_REF_1_1.fasta
    |__NGSphy_REF_1_2.fasta
    ...
  |__2/
    |__NGSphy_REF_2_1.fasta
    |__NGSphy_REF_2_2.fasta
    ...
```

7.6. Scripts

This folder will store all the scripts for ART execution, according to the options in the execution block. If we decide to run NGSphy for any cluster environment, we will have the job script and the seed file. If we choose bash as environment and we do not want to execute the ART commands within NGSphy, we would have a single bash script.

SGE	SLURM
reads/ |__project.sge.sh |__project.seedfile.txt	reads/ |__project.slurm.sh |__project.seedfile.txt
bash	
reads/ |__project.sh	

Where, **project** will be **NGSphy,** if using any of the single gene tree input modes. Whereas, for the gene-tree distribution input mode, it will be the name of the SimPhy output folder.

7.7. NGS mode

Data will be structured per replicate.

7.7.1. NGS reads ART

This folder will store the output of ART. It follows the same folder structure of the *individuals* folder, but instead of having FASTA files, it will contain the FASTQ files [and alignment and mapping files (ALN and SAM) if requested] generated by ART.

```
reads/
|__1/
  |__1/
    |__prefix_1_1_ind1_R1.fq
    |__prefix_1_1_ind1_R2.fq
    |__prefix_1_1_ind2_R1.fq
    |__prefix_1_1_ind2_R2.fq
  |__2/
    |__prefix_1_2_ind1_R1.fq
    |__prefix_1_2_ind1_R2.fq
    |__prefix_1_2_ind2_R1.fq
    |__prefix_1_2_ind2_R2.fq
|__2/
```

```
  |__1/
    |__prefix_2_1_ind1_R1.fq
    |__prefix_2_1_ind1_R2.fq
    |__prefix_2_1_ind2_R1.fq
    |__prefix_2_1_ind2_R2.fq
  |__2/
    |__prefix_2_2_ind1_R1.fq
    |__prefix_2_2_ind1_R2.fq
    |__prefix_2_2_ind2_R1.fq
    |__prefix_2_2_ind2_R2.fq
```

7.7.2. NGS read counts

This folder will store the output obtained from the read count simulation. This folder is structured in 2 sub-folders (with and without sequencing errors), each structured per replicate, and containing as many VCF files as loci.

Sub-folders will be:

- **no_error:** VCF files with the simulated read counts without sequencing error.
- **with_error:** VCF files with the simulated read counts with the introduced sequencing error.

```
reads
  |__no_error/
    |__1/
      |__prefix_1_1_TRUE.VCF
      |__prefix_1_2_TRUE.VCF
      |__prefix_1_3_TRUE.VCF
    |__2/
      |__prefix_2_1_TRUE.VCF
      |__prefix_2_2_TRUE.VCF
      |__prefix_2_3_TRUE.VCF
  |__with_error/
    |__1/
      |__prefix_1_1.VCF
      |__prefix_1_2.VCF
      |__prefix_1_3.VCF
    |__2/
      |__prefix_2_1.VCF
      |__prefix_2_2.VCF
      |__prefix_2_3.VCF
```

7.8. Files

7.8.1. Running times file

Stores information related to the time used in each ART run or read-count thread per locus. This file will contain input/output files for each process and its corresponding individual, locus (gene-tree) and replicate (REPID). See more on Section 6.6.3

Example of the file

```
1,1,0,output/individuals/1/01/test_wrapper_1_01_data_0.fasta, 0.013984,
1479977980,output/reads/1/01/test_wrapper_1_01_data_0_R
1,1,1,output/individuals/1/01/test_wrapper_1_01_data_1.fasta, 0.014757,
1479977980,output/reads/1/01/test_wrapper_1_01_data_1_R
1,1,2,output/individuals/1/01/test_wrapper_1_01_data_2.fasta, 0.013589,
1479977980,output/reads/1/01/test_wrapper_1_01_data_2_R
1,1,3,output/individuals/1/01/test_wrapper_1_01_data_3.fasta, 0.013404,
1479977980,output/reads/1/01/test_wrapper_1_01_data_3_R
1,1,4,output/individuals/1/01/test_wrapper_1_01_data_4.fasta, 0.013775,
1479977980,output/reads/1/01/test_wrapper_1_01_data_4_R
```

7.8.2. Debug file

For each NGSphy run is optional to get a debug log file. If the **“-l/--log”** option in the command line is set to DEBUG, the file will be generated in the current working directory and under the name:

```
NGSPHY.YYYYMMDD-HH:mm:SS.log
```

- YYYY: year
- MM: month
- DD: day
- HH: hours
- mm: minutes
- ss: seconds

This file stores information of the program execution, at a very detailed level. A debug log file will look like this:

```
13/08/2017 11:21:19 AM - ERROR (__main__|handlingCmdArguments:82):     Something
happened while parsing the arguments.
Please verify. Exiting.
```

8. Additional information

8.1. Motivation

Advances in sequencing technologies have now made very common that datasets for phylogenomic inference consist of large numbers of loci from multiple species and individuals. The use of next-generation sequencing (NGS) for phylogenomics implies a complex computational pipeline where multiple technical and methodological decisions are necessary that might influence the final tree obtained, from coverage to assembly, mapping, variant calling and/or phasing. In order to assess the influence of these variables, here we introduce NGSphy, an open-source tool for the genome-wide simulation of Illumina reads obtained from thousands of gene families evolving under a common species tree, with multiple haploid and/or diploid individuals per species, where sequencing coverage (depth) heterogeneity can be modelized across individuals and loci, including off-target loci and phylogenetic decay. Moreover, parameter values for the different replicates can be sampled from user-defined statistical distributions.

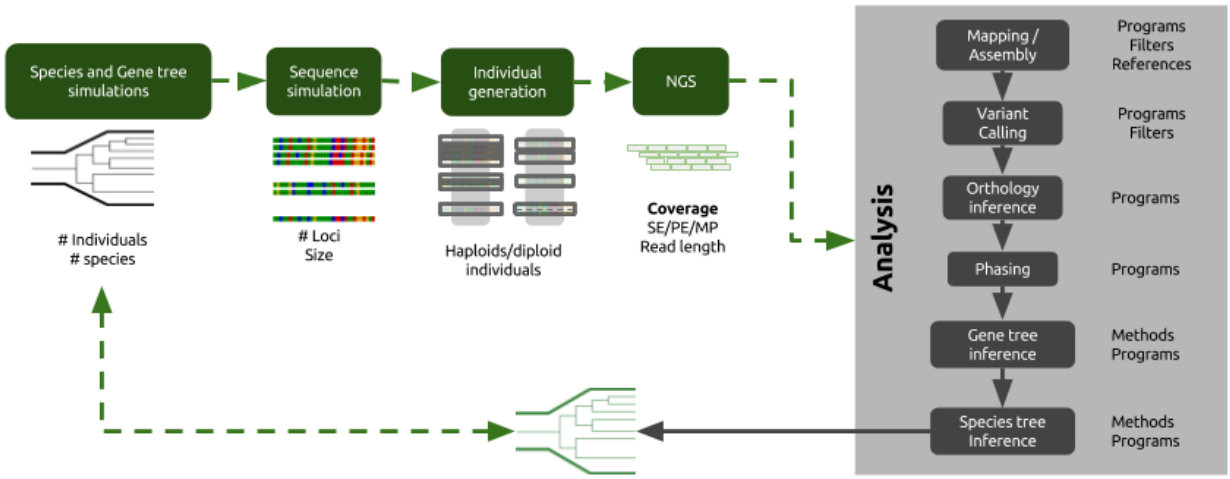


FIGURE 2: A possible analysis pipeline for multilocus, multispecies datasets with multiple individuals with the final goal of exploring the sensitivity of species tree inferences to NGS parameterization variation.

8.2. What can be done with NGSphy? The detailed scenarios

With NGSphy you can generate:

- haploid individuals from gene-tree distributions
- diploid individuals from gene-tree distributions
- genome sequences of haploid individuals from a single gene tree
- genome sequences of diploid individuals from a single gene tree
- genome sequences of haploid individuals from a single gene tree and an user-defined ancestral sequence

- genome sequences of diploid individuals from a single gene tree and an user-defined ancestral sequence
- NGS Illumina reads of haploid individuals
- NGS Illumina reads of diploid individuals
- NGS read counts of haploid individuals
- NGS read counts of diploid individuals
- For the NGS data generation, variation of coverage due to the following:
    - variation across individuals and/or loci
    - targeted-sequencing effects
        - on/off target loci
        - on-target loci not captured
        - taxon-specific variation

8.5. Third-party software involved

8.5.3. ART

- Huang W, Li L, Myers JR, and Marth, GT (2012) ART: a next-generation sequencing read simulator. *Bioinformatics* 28 (4): 593-594

ART (http://www.niehs.nih.gov/research/resources/software/biostatistics/art/) is a set of simulation tools to generate synthetic next-generation sequencing reads. ART simulates sequencing reads by mimicking real sequencing process with empirical error models or quality profiles summarized from large recalibrated sequencing data. ART can also simulate reads using user own read error model or quality profiles. ART supports simulation of single-end, paired-end/mate-pair reads of three major commercial next-generation sequencing platforms: Illumina's Solexa, Roche's 454 and Applied Biosystems' SOLiD. ART can be used to test or benchmark a variety of method or tools for next-generation sequencing data analysis, including read alignment, *de novo* assembly, SNP and structural variation discovery. ART outputs reads in the FASTQ format, and alignments in the ALN format. ART can also generate alignments in the SAM alignment or UCSC BED file format.

8.5.2. INDELible

- William Fletcher and Ziheng Yang (2009) INDELible: A flexible simulator of biological sequence evolution. Molecular Biology and Evolution. 26 (8): 1879–88. doi:10.1093/molbev/msp098

INDELible (http://abacus.gene.ucl.ac.uk/software/indelible/) is an application for biological sequence simulation that combines many features. Using a length-dependent model of indel formation it can simulate evolution of multi-partitioned nucleotide, amino-acid, or codon data sets through the processes of insertion, deletion, and substitution in continuous time.

Nucleotide simulations may use the general unrestricted model or the general time reversible model and its derivatives, and amino-acid simulations can be conducted using fifteen different

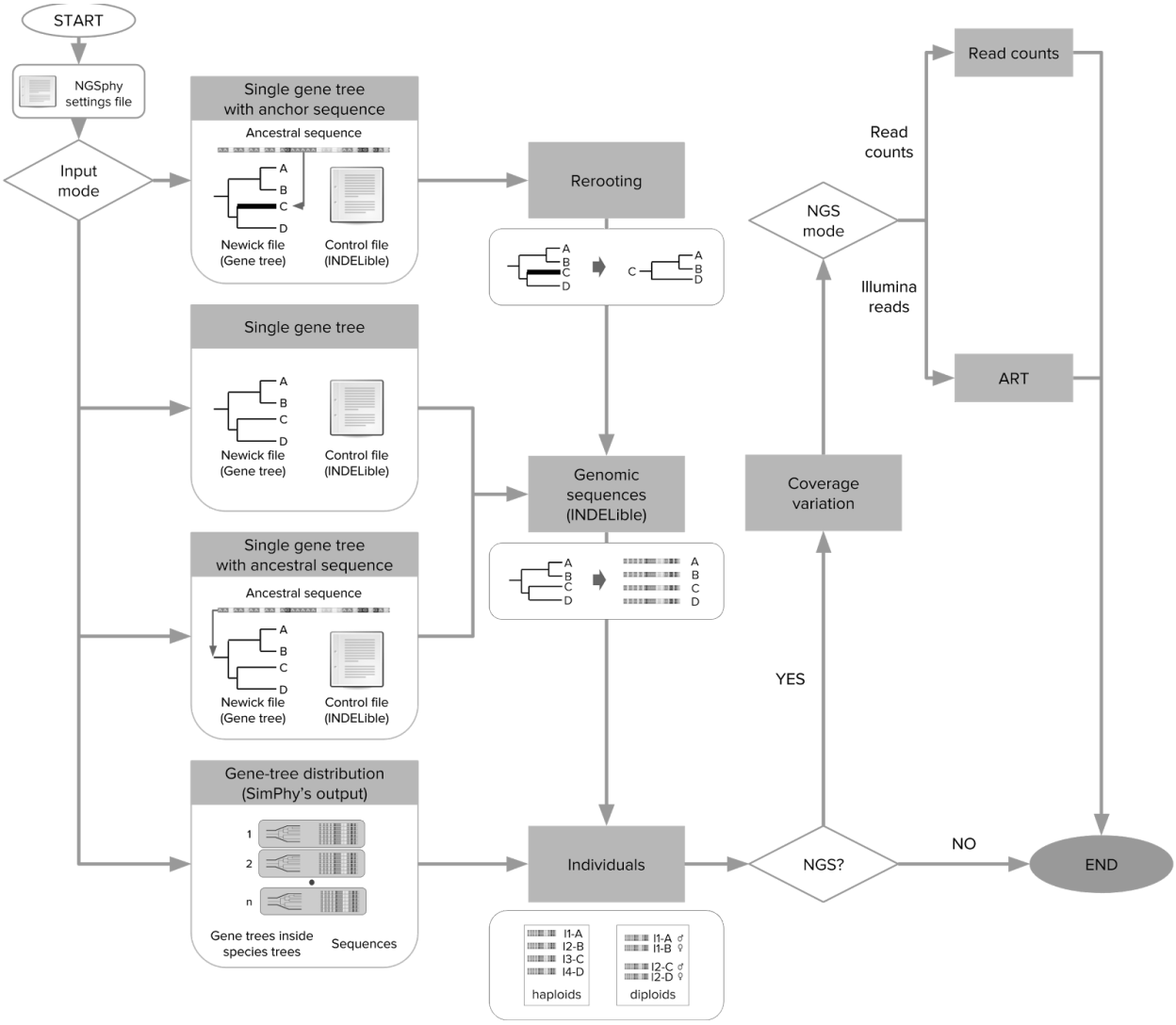
empirical rate matrices. Substitution rate heterogeneity can be modelled via the continuous and discrete gamma distributions, with or without a proportion of invariant sites. INDELible can also simulate under non-homogenous and non-stationary conditions where evolutionary models are permitted to change across a phylogeny. Unique among indel simulation programs, INDELible offers the ability to simulate using codon models that exhibit nonsynonymous/synonymous rate ratio heterogeneity among sites and/or lineages.

8.5.1. SimPhy

- Diego Mallo, Leonardo De Oliveira Martins and David Posada (2015). SimPhy : Phylogenomic Simulation of Gene, Locus, and Species Trees. Systematic Biology., November, syv082. doi:10.1093/sysbio/syv082

SimPhy (https://github.com/adamallo/simphy) is a program for the simulation of gene family evolution under incomplete lineage sorting (ILS), gene duplication and loss (GDL), replacing horizontal gene transfer (HGT) and gene conversion (GC). SimPhy simulates species, locus and gene trees with different levels of rate heterogeneity, and uses INDELible to evolve nucleotide/codon/aminoacid sequences along the gene trees. The input for SimPhy are the simulation parameter values, which can be fixed or sampled from user-defined statistical distributions. The output consists of sequence alignments and a relational database that facilitate posterior analyses.

8.6. NGSphy workflow



NGSphy, verifies all the content of the project, the settings files involved and/or the existence of the corresponding third-party applications in order to run. If the input data corresponds to the single gene tree an user-defined ancestral sequence, first the tree is rooted to the selected gene-tree tip. The next step (for any single gene tree input mode) is to evolve the tree under the specific evolution mode to obtain the expected genome sequences. Then (any input mode), the *generation of individuals*, whether haploid or diploid:

- For haploid individuals, resulting genome sequences are separated into single FASTA files and identified. In addition, a file is generated with the correspondence between the individual generated and the description of the sequence it belongs to.
- For diploid individuals, there is a process of verification that the project content includes species-trees with an even number of individuals per taxa. Sequences are then "paired", individuals being generated by randomly sampling without replacement two sequences

within the same gene family and species. Output will include a table for each replicate with the identifiers for the sequences paired and the individuals generated.

Afterwards, the coverage variation matrices will be computed according to the parameters introduced (detailed process here) and finally the sequencing data generated, consist on either Illumina reads or read counts (VCF files).

- For the Illumina reads, program calls out ART, the NGS simulator, with the parameters established in the settings file and generates reads from the previously generated individuals. Resulting files depend on the settings introduced, and they are files related to the execution of the ART processes (scripts and text files), and the output of such processes (ALN, BAM and/or FASTQ files).
- For read counts, two scenarios are simultaneously computed, with and without errors.

8.7. Read count simulation

The read count approach is based on the assumption (Ritz et al., 2011) that the sequencing process is uniform in generating short reads from the target genome, and that the number of reads mapped to a region is expected to be proportional to the number of times the region appears in a DNA sample (Ji and Chen, 2015). Read counts are produced under a user-defined error rate. First, the variable sites (regarding the reference sequences) are identified. Then, coverage for each position is sampled from a Negative Binomial distribution whose mean and overdispersion parameter are the sampled coverage for the specific locus and individual. For diploid individuals, coverage is further splitted among chromosomes with equal probability. Genotype likelihoods for every site are computed as in GATK (McKenna et al 2010) (see also Korneliussen et al. 2014). The output is a set of VCF files, one per locus.

9. Getting help

Most common issues, doubts and questions should be solved by reading this manual. If that is not the case or you find any bug, you can post an issue to this repository for reproducibility purposes, with the following files attached:

- the settings file
- **<simphy_project_name>.command** file or the **indelible_control.txt** file.

10. Development and testing

This software has been developed for Linux/Mac environments and specifically tested under:

- Linux Kernel:

```
4.8.0-58-generic #63~16.04.1-Ubuntu SMP Mon Jun 26 18:08:51 UTC 2017 x86_64
x86_64 x86_64 GNU/Linux
```

- Distribution:

```
Ubuntu 16.04.2 LTS
```

- Hardware:

```
Dual core Intel Core i5-3427U (-HT-MCP-) cache: 3072 KB
8GB RAM
```

References

- (Huang et al. 2012) Weichun Huang, Leping Li, Jason R Myers, and Gabor T Marth (2012) ART: a next-generation sequencing read simulator. Bioinformatics 28 (4): 593-594
- (Fletcher and Yang 2009) William Fletcher and Ziheng Yang (2009) INDELible: A flexible simulator of biological sequence evolution. Molecular Biology and Evolution. 26 (8): 1879–88. https://doi.org/10.1093/molbev/msp098
- (Mallo et al. 2015) Diego Mallo, Leonardo De Oliveira Martins and David Posada (2015). SimPhy : Phylogenomic Simulation of Gene, Locus, and Species Trees. Systematic Biology., November, syv082. https://doi.org/10.1093/sysbio/syv082
- (Korneliussen et al., 2014) Thorfinn Sand Korneliussen, Anders Albrechtsen and Rasmus Nielsen (2014) ANGSD: Analysis of Next Generation Sequencing Data. BMC Bioinformatics201415:356. https://doi.org/10.1186/s12859-014-0356-4
- (McKenna et al. 2010) McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., ... & DePristo, M. A. (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome research, 20(9), 1297-1303.
- (Godden et al. 2012) Godden, G. T., Jordon-Thaden, I. E., Chamala, S., Crowl, A. A., García, N., Germain-Aubrey, C. C., ... & Gitzendanner, M. A. (2012). Making next-generation sequencing work for you: approaches and practical considerations for marker development and phylogenetics. Plant Ecology & Diversity, 5(4), 427-450.
- (McTavish et al. 2017) Emily Jane McTavish, James Pettengill, Steven Davis, Hugh Rand, Errol Strain, Marc Allard and Ruth E. Timme (2017) TreeToReads - a pipeline for simulating raw reads from phylogenies. BMC Bioinformatic 201718:178 https://doi.org/10.1186/s12859-017-1592-1
- (Mallo and Posada 2016) Mallo, D., & Posada, D. (2016). Multilocus inference of species trees and DNA barcoding. Phil. Trans. R. Soc. B, 371(1702), 20150335.
- (Ogilvie et al. 2016) Ogilvie, H. A., Heled, J., Xie, D., & Drummond, A. J. (2016). Computational performance and statistical accuracy of* BEAST and comparisons with other methods. Systematic biology, 65(3), 381-396.
- (Carstens et al. 2013) Bryan C. Carstens, Tara A. Pelletier, Noah M. Reid, Jordan D. Satler (2013) How to fail at species delimitation. Molecular Ecology. Volume 22, Issue 17. 4369–4383
- (McCormack et al. 2013) McCormack JE, Hird SM, Zellmer AJ, Carstens BC, Brumfield RT (2013) Applications of next-generation sequencing to phylogeography and phylogenetics. Molecular Phylogenetics and Evolution, 66, 526–538.
- (Mardis 2008) Mardis, E. R. (2008). The impact of next-generation sequencing technology on genetics. Trends in genetics, 24(3), 133-141.
- (Schuster 2008) Schuster, S. C. (2008). Next-generation sequencing transforms today's biology. Nature methods, 5(1), 16.
- (Reis-Filho 2009) Reis-Filho, J. S. (2009). Next-generation sequencing. Breast Cancer Research, 11(3), S12.
- (Xiong et al 2011) Xiong, M., Zhao, Z., Arnold, J., & Yu, F. (2011). Next-generation sequencing. Journal of BioMed Research, 2010.
- (Koboldt et al 2013) Daniel C.Koboldt , Karyn Meltz Steinberg, David E.Larson, Richard K.Wilson Elaine R.Mardis (2013) The Next-Generation Sequencing Revolution and Its Impact on Genomics. Cell. Volume 155, Issue 1, 26 September 2013, Pages 27-38 https://doi.org/10.1016/j.cell.2013.09.006
- (van Dijk et al 2016) Erwin L.van Dijk, Hélène Auger, Yan Jaszczyszyn adn Claude Thermes (2016) Ten years of next-generation sequencing technology. Trends in Genetics. Volume 30, Issue 9, September 2014, Pages 418-426 https://doi.org/10.1016/j.tig.2014.07.001
- (Goodwin et al 2016) Sara Goodwin, John D. McPherson and W. Richard McCombie (2016) Coming of age: ten years of next-generation sequencing technologies. Nature Reviews Genetics 17, 333–351 https://doi.org/doi:10.1038/nrg.2016.49
- (Bertels et al. 2014) Frederic Bertels Olin K. Silander Mikhail Pachkov Paul B. Rainey Erik van Nimwegen (2014) Automated Reconstruction of Whole-Genome Phylogenies from Short-Sequence Reads. Molecular Biology and Evolution. 31 (5): 1077-1088. DOI: https://doi.org/10.1093/molbev/msu088
- (Catchen et al. 2013) Catchen, J., Hohenlohe, P. A., Bassham, S., Amores, A., & Cresko, W. A. (2013). Stacks: an analysis tool set for population genomics. Molecular ecology, 22(11), 3124-3140.
- (Giolai et al. 2017) Michael Giolai, Pirita Paajanen, Walter Verweij, Kamil Witek, Jonathan D. G. Jones and Matthew D. Clark (2017) Comparative analysis of targeted long read sequencing approaches for characterization of a plant's immune receptor repertoire. BMC Genomics 2017 18:564. https://doi.org/10.1186/s12864-017-3936-7

- (Escalona et al. 2016) Merly Escalona, Sara Rocha and David Posada (2016) A comparison of tools for the simulation of genomic next-generation sequencing data. Nature Reviews Genetics 17, 459–469 (2016) doi:10.1038/nrg.2016.57
- (McTavish et al. 2016) Emily Jane McTavish, James Pettengill, Steven Davis, Hugh Rand, Errol Strain, Marc Allard and Ruth E. Timme (2016) TreeToReads - a pipeline for simulating raw reads from phylogenies. BMC Bioinformatics 18:178. https://doi.org/10.1186/s12859-017-1592-1
- (Huelsenbeck 1995) John P. Huelsenbeck (1995) Performance of Phylogenetic Methods in Simulation. Syst Biol (1995) 44 (1): 17-48. DOI: https://doi.org/10.1093/sysbio/44.1.17
- (Beaumont et al. 2002) Beaumont, M. A., Zhang, W., & Balding, D. J. (2002). Approximate Bayesian computation in population genetics. Genetics, 162(4), 2025-2035.
- (Linder et al. 2010) Linder, C. R., Suri, R., Liu, K., & Warnow, T. (2010). Benchmark datasets and software for developing and testing methods for large-scale multiple sequence alignment and phylogenetic inference. PLoS currents, 2.
- (Mamanova et al. 2010) Mamanova L, Coffey AJ, Scott CEet al.(2010) Target-enrich-ment strategies for next- generation sequencing.Nature Meth-ods,7, 111–118
- (Bi et al. 2012) Bi K, Vanderpool D, Singhal S, Linderoth T, Moritz C, GoodJM (2012) Transcriptome based exon capture enables highlycost-effective comparative genomic data collection at moder-ate evolutionary scales.BMC Genomics,13, 403
- (Lemmon et al. 2012) Lemmon AR, Emme SA, Lemmon EM (2012) Anchored hybridenrichment for massively high-throughput phylogenomics.Systematic Biology,61, 727–744.
- (Faircloth et al. 2012) Faircloth BC, McCormack JE, Crawford NG, Harvey MG,Brumfield RT, Glenn TC (2012) Ultraconserved elementsanchor thousands of genetic markers spanning multiple evo-lutionary timescales.Systematic Biology,61, 717–726
- (Baird et al. 2008) Baird NA, Etter PD, Atwood TSet al.(2008) Rapid SNP discov-ery and genetic mapping using sequenced RAD markers.PLoS One,3, e3376
- (Davey et al. 2011) Davey JW, Hohenlohe PA, Etter PD, Boone JQ, Catchen JM,Blaxter ML (2011) Genome-wide genetic marker discoveryand genotyping using next-generation sequencing.NatureReviews Genetics,12, 499–510.
- (Ji and Chen 2015) Tieming Ji and Jie Chen (2015) *Modeling the next generation sequencing read count data for DNA copy number variant study*. Stat. Appl. Genet. Mol. Biol. 2015; 14(4): 361–374. https://doi.org/10.1515/sagmb-2014-0054
- (Liao et al. 2014) Yang Liao Gordon K. Smyth Wei Shi (2014) *featureCounts: an efficient general purpose program for assigning sequence reads to genomic features.* Bioinformatics (2014) 30 (7): 923-930. https://doi.org/10.1093/bioinformatics/btt656
- (Bragg et al. 2016) Bragg, J. G., Potter, S., Bi, K., & Moritz, C. (2016). Exon capture phylogenomics: efficacy across scales of divergence. Molecular ecology resources, 16(5), 1059-1068.