GlbPSs 1.0 Manual Part 5: Documentation of indloc_22.0.pl

5/22/2015 Andreas Hapke,
Institute of Anthropology, Johannes Gutenberg University Mainz, Germany, ahapke2@gmail.com

This program identifies loci and alleles and determines their sequencing depths within individuals.

Algorithm overview

The program analyzes sequence data from different individuals separately. It collapses identical reads into sequence variants (*svars*) with a given depth (number of reads). Good *svars* have a configurable minimum depth (-D). Variants with smaller depths are rare *svars*.

Locus identification

indloc uses two distance criteria, *good_svar_d* and *rare_svar_d*, which you can specify for different sequence lengths. The identification of loci starts with the good *svars*. The program performs pairwise comparisons between good *svars* of the same length and identifies pairs with a distance ≤ *good_svar_d*. The program then builds networks of good *svars* wherein all edges have a length ≤ *good_svar_d*. Each network founds the basis for an individual locus. Good *svars* that are not part of any pair found separate loci.

Next, the algorithm adds rare *svars* to existing loci. It compares each rare *svar* to all existing loci of the same sequence length. If the distance between a rare *svar* and at least one good *svar* of a locus is ≤ *rare_svar_d*, that locus becomes a candidate locus. The algorithm assigns a rare *svar* to a locus, when it is the only candidate locus.

SNP calling and error correction

You can select one of two methods for SNP calling: a frequency threshold method and a binomial likelihood ratio method (Glaubitz *et al.* 2014). For each locus, the program builds a matrix of all *svars* with sequences as rows and positions as columns. It evaluates each variable position with the selected SNP calling method, accepts none, one or more characters as valid and treats all other characters as errors. The program always treats N as error. When there is no valid character, it changes all characters to N. When there is one valid character, it corrects all error characters to the single valid one. When there is more than one valid character, it changes all others to N. Under the frequency threshold method, you can also instruct the program to treat all characters except N as valid or to perform no error correction at all.

Identification of alleles

Next, the program performs pairwise distance comparisons between all *svars* of a locus again and builds networks of *svars* with a connecting distance of 0 (treating N as identical with ACGT). If not all pairwise distances in a network are 0 (which can happen when a *svar* contains N), the program splits up the network. All pairwise distances in each resulting network are 0. Each network builds the basis for an allele. The program determines the sequence of the allele by a second correction step based on the matrix of all *svars* of this allele: If a column contains only one character (NACGT), this character appears in the sequence of the allele at that position. If a column contains N (an erroneous character) and one valid character (ACGT) the algorithm selects the valid character for that position.

Sequencing depths of alleles

The depth of a locus is the total number of reads assigned to it. The depth of an allele is determined during the identification of alleles: In most cases, the depth of an allele is the total depth of its constituting *svars*. A special case occurs when the algorithm splits up a network during allele identification. *Svars* containing N are discarded during this step. The algorithm adds depth from certain discarded *svars* to retained *svars* (see algorithm details).

Filtering of alleles

The program discards all alleles that contain any N in their sequence. If you use the frequency threshold method for SNP calling, you can further instruct the algorithm to retain only alleles with a minimum fraction of the total read depth (allele frequency threshold). If you use the binomial likelihood ratio method, the program accepts the two deepest retained alleles or more alleles in special cases (see Command flags in detail: Binomial likelihood ratio method).

Ambiguous genotypes: more than two alleles, conflicts between SNPs

It can happen that **indloc** identifies more than two alleles at an individual locus. Both SNP calling methods can accept more than two characters at a position. Moreover, several SNPs at a locus may not be in phase and define more than two, possibly many alleles. It is further possible that **indloc** finally accepts only one or two of these alleles, which represent a small fraction of the total read depth. Such situations arise when there is a high sequencing/polymerase error rate at a locus/position or when **indloc** assigned paralogous sequences to the same locus. You can later identify and remove such ambiguous genotypes. The program **depth_analyzer** helps to detect loci with extreme sequencing depth, which could be repetitive elements. The program **data_selector** enables you to remove these loci. **data_selector** offers many additional filters. You can remove loci that have more than two alleles in any individual or remove individual genotypes with more than two alleles. You can further select only those genotypes that represent a given fraction of the total read depth by the accepted alleles. These filters enable you to exclude conflicts between called SNPs in all selected genotypes of your dataset.

Usage

```
indloc_22.0.pl
```
or
```
perl indloc_22.0.pl
```

Several command flags enable you to control the analysis. Example:

```
indloc_22.0.pl –i ind_fastqin.txt –d distance_file.txt –D 6 –P 3 –c 0.2 –a 0.2
```

Command flags (default values in parentheses)

-i	(indloc_infiles.txt)	Name (or path) of file ind_infiles
-z	(0)	1: Produce gzip compressed outfiles *_svars.txt.gz and *_reads.txt.gz 0: Produce plain text outfiles *_svars.txt and *_reads.txt
-d	(none)	Name (or path) of file distance_settings
-D	(4)	Minimum depth of good sequence variants (positive integer)
-P	(1)	Maximum number of processes in parallel (positive integer)
-r	(0)	1: Build r-read of rc-duplicates, 0: do not
-M	(f)	Method for SNP calling (f: frequency threshold, b: binomial likelihood ratio)
Flags for method f		
-c	(0.2)	Character frequency threshold (-1 or decimal between 0 and 1)
-a	(0.2)	Allele frequency threshold (decimal between 0 and 1)
Flag for method b		
-e	(0.01)	Sequencing error rate (decimal >= 0 and <1).

Command flags in detail:

-i File ind_infiles

Format: Text file with one line per individual: individual ID TAB name (or full path) of a fastq-infile. There must be one fastq-infile per individual. Example:

```
ind_01      C:\my_fastq_files\fdm_run_1\TTCT_R1.fq
ind_02      C:\my_fastq_files\fdm_run_1\GCTCTA_R1.fq
```

The fastq-infiles may be gzip compressed files or plain text files - depending on your settings for **fdm**, the previous program. Names of gzip compressed files must have the extension ".gz". Plain text files may have any other or no extension. Each gzip compressed file must contain one single fastq-infile. **indloc** cannot unpack a tarball. When you have produced your fastq-infiles with **fdm**, they will meet all these conditions.

Fastq-infiles must have this format: Text file with four lines per sequence: Header, sequence, third line, phredsymbols. No additional blank lines. Allowed characters in the sequence: NACGT (capitals). Sequences may have different lengths.

-z Produce gzip compressed outfiles

indloc produces five outfiles for each individual (see below). Per default (`-z 0`) all outfiles are plain text files. Two files for each individual are relatively large: `*_svars.txt` and `*_reads.txt`. If you wish to save disk space, use `-z 1`. The program then produces gzip compressed files: `*_svars.txt.gz` and `*_reads.txt.gz`. This requires additional time. The *_reads.txt files are needed by **pair_finder**. This program automatically determines the infile type and reads in plain text files or gzip compressed files. Again, reading of gzip compressed files by **pair_finder** requires more time than reading of plain text files.

-d File distance_settings

Format: Text file with three integers per line separated by TABs:

sequence length
good_svar_d
rare_svar_d

Example:

```
30    3     4
60    4     6
```

The variable *good_svar_d* defines the maximum distance allowed between neighboring good sequence variants (good *svars*) in a locus network. Two good *svars* with a distance ≤ *good_svar_d* are assigned to the same locus. The variable *rare_svar_d* defines the maximum distance allowed between a rare sequence variant (rare *svar*) and at least one good *svar* in a locus network. You can define different values for different sequence lengths. It is not necessary to define values for all sequence lengths occurring in your fastq-infiles. With the example file above, **indloc** will use "*good_svar_d* 3 *rare_svar_d* 4" for all sequences with lengths from 1 to 59 and "*good_svar_d* 4 *rare_svar_d* 6" for all sequences with length 60 and greater. If you do not provide the name of a distance settings file, **indloc** uses "*good_svar_d* 4 and *rare_svar_d* 6" for all sequence lengths.

-D minimum depth of good sequence variants

Sequence variants with smaller depth are rare *svars*.

-P maximum number of processes in parallel

The program analyzes each individual fastq infile separately. `-P 3` means that up to 3 fastq infiles are analyzed in parallel. I recommend that you start with a small value and monitor the memory usage before you use greater values.

-r Build r-read of rc-duplicates

You can build *rc-duplicates* with **fdm** and analyze them with **indloc**. To save disk space, you can instruct **fdm** to save only the f-reads of your *rc-duplicates*. If you have done so, use `-r 1` in **indloc**. This instructs **indloc** to read in the f-reads and to create the corresponding r-reads, which are the reverse complements of the f-reads. Please make sure that your fastq infiles contain only f-reads of *rc-duplicates* in this case - see the documentation of **fdm** for details. All fastq headers must end with "_1" - use `ch_h 1` in **fdm** to achieve that. Headers of reverse reads created by **indloc** will end with "_2". Apart from saving disk space, you may also save a bit of time. I observed that **indloc** needed less time to construct the r-reads than to read them in from a combined fastq file.

In all other cases use the default `-r 0`.

-M method for SNP calling

Type `-M f` to use the frequency threshold method or `-M b` to use the binomial likelihood ratio method.

Frequency threshold method for SNP calling

A character frequency threshold (command flag -c) controls SNP calling with this method. The algorithm analyses each variable position in an individual locus alignment of *svars*. At a given position, it determines the sequencing depth (frequency) of each character and identifies valid characters. With a threshold of `-c 0.2`, valid characters must have a minimum depth corresponding to 20% of the total number of reads in the locus. The algorithm rounds this minimum depth down. Examples:

`-c 0.2` total number of reads in the locus: 8, minimum sequencing depth: 1.
`-c 0.2` total number of reads in the locus: 17, minimum sequencing depth: 3.

The algorithm treats other characters as error. It always treats N as error. The algorithm changes all characters to N, when there is no valid character. When there is one valid character, it corrects all error characters to the single valid one. When there is more than one valid character, it changes all others to N.

You can enter `-c 0` to treat all characters except N as valid or `-c -1` to inactivate any error correction.

-a allele frequency threshold

This variable controls the optional filtering of alleles after calling SNPs with the frequency threshold method. Frequency here means the sequencing depth of an allele compared to the total number of reads in the locus including discarded reads. After having collapsed all corrected *svars* into potential alleles, the program discards all alleles containing N. (An allele contains N when all its reads contain N at a specific position.) Enter `-a 0` to perform no additional filtering. Select a threshold between 0 and 1 to accept only alleles with a minimum depth corresponding to your threshold. The algorithm rounds the minimum acceptable depth down. Examples:

`-a 0.2` total number of reads in a locus 8, minimum depth: 1. The algorithm will discard alleles containing N and retain all other alleles.

`-a 0.2` total number of reads in a locus 17, minimum depth: 3. The algorithm will discard alleles containing N and alleles with a depth below 3.

Binomial likelihood ratio method for SNP calling (Glaubitz *et al.* 2014)

A configurable sequencing error rate *e* (command flag -e) controls SNP calling with this method. The algorithm analyses each variable position in an individual locus alignment of *svars*. At a given position, it determines the sequencing depth (count) of each character except N and sorts the values in descending order: *p1* ≥ *p2* ≥ *p3* ≥ *p4*. It then uses *p1*, *p2* and the sequencing error rate *e* to decide if it accepts a SNP based on the binomial likelihood ratio

$$\frac{p_{Het}}{p_{Hom}} = \frac{0.5^{p1+p2}}{(1-e)^{p1}e^{p2}}$$

where p_{Het} is the binomial probability of *p1* and *p2* under the assumption that the individual is heterozygous, and p_{Hom} is the binomial probability of *p1* and *p2* under the assumption that it is homozygous and all observed characters with *p2* are sequencing errors, which occur at rate *e*. The algorithm accepts a SNP when the ratio is > 1. In the special case of *e* = 0, it always accepts a SNP when there are at least 2 characters other than N. When the algorithm does not accept a SNP, it treats the character with *p1* as valid and all others as errors. When it accepts a SNP, it treats the characters with *p1* and *p2* as valid. Further characters are valid in the special cases where *p2* = *p3* and *p2* = *p4*. The algorithm treats other characters as error. It always treats N as error. When there is one valid character, it corrects all error characters to the single valid one. When there is more than one valid character, it changes all others to N.

Filtering of alleles under the binomial likelihood ratio method for SNP calling

After having collapsed all corrected *svars* into potential alleles, the program discards all alleles containing N. (An allele contains N when all its reads contain N at a specific position.) Next, the algorithm selects the two alleles with the greatest depths and discards all other alleles. It retains more than two alleles in the special case where two or more alleles have the same second highest depth.

Outfiles

The program produces a file `indloc_report.txt`, a file `individuals.txt` and five files for each individual: `ind_01_indloc_report.txt`, `ind_01_loci.txt`, `ind_01_alleles.txt`, `ind_01_svars.txt` and `ind_01_reads.txt` where "ind_01" is the ID of this individual.

`indloc_report.txt`

This file contains the settings used by the program, the IDs of the analyzed individuals, the runtime, and notes about successfully completed steps or error messages (if any). If a file `indloc_report.txt` already exists in the directory containing the program, the new report is appended to the existing file.

`individuals.txt`

This file contains a list of individual IDs of the analyzed individuals as specified in infile `ind_infiles`. If a file `individuals.txt` already exists, the program reads in the existing file and compares the IDs of the new individuals to those in the existing file. If an individual ID already exists, the program produces an error message and stops execution. If this is not the case, the program appends the new individual IDs to the existing file `individuals.txt`.

`ind_01_indloc_report.txt`

This file contains the text "analysis completed" or error messages for this individual.

`ind_01_loci.txt`

Format: tab-delimited text table with header line. One line per individual locus:

Loc_ID: ID of this individual locus

Loc_cat: “lost” or “valid”: Lost means that all identified alleles have been discarded due to N in the sequence or low frequency. Valid means that at least one allele without N was retained.

seql: Sequence length

Loc_dep: Total read depth of the locus including discarded reads

n_alleles: Number of accepted alleles

lost_alleles: Number of putative alleles lost from the locus during identification of alleles and filtering of alleles.

potlostmore: 1/0: 1 if the number of lost alleles could be greater than *lost_alleles*

`ind_01_alleles.txt`

Format: tab-delimited text table with header line. One line per individual allele:

Loc_ID: ID of this individual locus

Loc_cat: “lost” or “valid”: Lost means that all identified alleles have been discarded due to N in the sequence or low frequency. Valid means that at least one allele without N was retained.

seql: Sequence length

All_ID: ID of this individual allele

Allseq: Allele sequence

Alldep: Depth of this allele. In most cases, the total depth of the constituting *svars*. In some cases, the depth of an allele includes additional depth from discarded *svars* (see Algorithm details).

`ind_01_svars.txt`

Format: tab-delimited text table with header line. One line per sequence variant (*svar*):

Loc_ID: ID of this individual locus

Loc_cat: “lost” or “valid”: Lost means that all identified alleles have been discarded due to N in the sequence or low frequency. Valid means that at least one allele without N was retained.

seql: Sequence length

All_ID: ID of this individual allele

svarID: ID of this *svar*

svarcat: “used” for *svars* that have been assigned to a retained allele, “discarded” for discarded *svars*.

svarseq: Original (uncorrected) sequence of this *svar*.

svardep: read depth of this *svar*

`ind_01_reads.txt`

Format: tab-delimited text table with header line. One line per read:

Loc_ID: ID of this individual locus

Loc_cat: “lost” or “valid”: Lost means that all identified alleles have been discarded due to N in the sequence or low frequency. Valid means that at least one allele without N was retained.

seql: Sequence length

All_ID: ID of this individual allele

svarID: ID of this *svar*

svarcat: “used” for *svars* that have been assigned to a retained allele, “discarded” for discarded *svars*.

readID: fastq-header of this read

Algorithm details

Collapsing of identical reads into sequence variants (svars)

The algorithm treats N as different from ACGT in this step. It sorts reads with different lengths into separate subsets of data and analyzes them separately.

Locus identification: pairwise distances

The identification of loci involves a great number of pairwise sequence comparisons in two steps: First, the program identifies pairs of good *svars* with a distance ≤ *good_svar_d*. In the second round, it identifies pairs of one rare and one good *svar* with a distance ≤ *rare_svar_d*.

The algorithm calculates distances as number of differing nucleotides. It treats N as identical with ACGT. Instead of calculating the pairwise distance for each *svar* pair (which would require extensive computation time), the algorithm first identifies pairs of *svars* that *could* have a pairwise distance ≤ d_{max} (where d_{max} is *good_svar_d* or *rare_svar_d*) and then calculates distances only for these candidate pairs. The algorithm identifies candidate pairs by splitting each *svar* into d_{max} +1 nonoverlapping fragments at constant positions. If a fragment contains one or more N characters, the algorithm builds all possible permutations of the fragment by changing N to A, C, G, and T. The algorithm sorts all fragments into a hash data structure and identifies pairs of *svars* that have at least one identical fragment at the same position – the candidate pairs.

Sequencing and polymerase error correction

Sequence variants can contain unknown characters (N). The SNP calling algorithms can further identify erroneous characters, which result from polymerase and sequencing error. I use the frequency threshold method for the following example (Fig. 1). The correction procedure is the same under the binomial likelihood ratio method. A locus comprises four *svars* ABCD with a total depth of 13. The character frequency threshold (-c) is 0.2, which evaluates to a minimum depth of 2.6 for valid characters. The algorithm rounds the minimum depth down to 2.

svar	depth	original seq.	corrected seq.
A	6	AAC	AAC
B	1	**T**AC	AAC
C	5	AGT	AGT
D	1	**N**G**G**	AG**N**

Fig. 1 First step of error correction. Only variable positions are shown. Characters treated as error are written in bold.

Characters N and T in column 1 are error characters (T because of its small depth). The algorithm corrects both erroneous characters to the single valid one: A. Colum 3 contains two valid characters, C and T, and one error character: G. The algorithm changes G to N.

Identification of alleles

The program calculates pairwise distances between all *svars* of the locus treating N as equal to ACGT and clusters *svars* into networks with connecting distances of 0. Each networks builds the basis for one allele. Figure 2 shows the resulting groups of *svars* for the example from Fig. 1.

allele	svar	depth	sequence
1	A	6	AAC
1	B	1	AAC
2	C	5	AGT
2	D	1	AG**N**

Fig. 2 Clustering of *svars* into networks with a connecting distance of 0. Characters treated as error are written in bold.

The identification of allele sequences involves a second error correction step. All *svars* of one allele are stored in a matrix. If a column contains N and another character, N is corrected to that character. If

it contains only N, no correction happens. All *svars* finally have the same sequence, which is the allele sequence. In most cases, as in our example here, the depth of an allele is the sum of depths of its *svars*:

```
Allele      depth       sequence

1           7           AAC
2           6           AGT
```

Fig. 3 Identification of allele sequences and depths

Identification of alleles: promiscuous networks

In rare cases, it can happen that a *svar* containing N has distance 0 to two different *svars*, which have a pairwise distance >0. In the resulting network, not all pairwise distances are 0. I call such networks *promiscuous networks* here (Fig. 4).

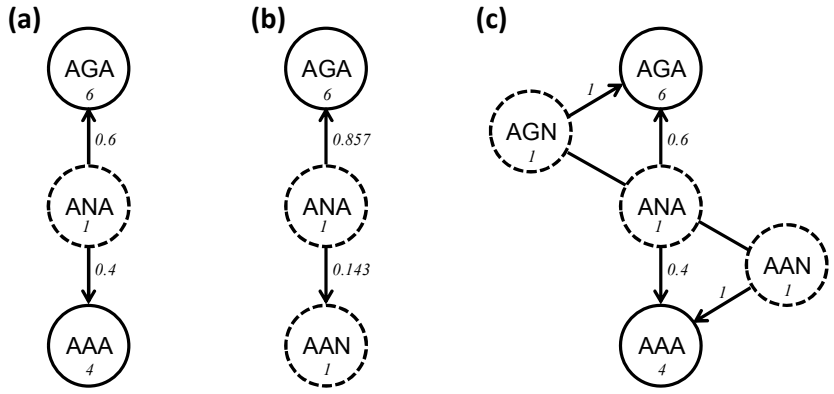

Fig. 4 Treatment of promiscuous networks under the frequency method for SNP calling. Dashed lines encircle *svars* containing N; solid lines encircle retained *svars*. Numbers in circles depict *svar* depths. Arrows symbolize donation of depth. Numbers at errors depict donated depth.

The algorithm splits up such networks by discarding all *svars* containing N. It does not always discard the depth of discarded *svars*. Retained *svars* (receivers) can receive some additional depth from those discarded *svars* (donors) that have a distance of 0 to them. If you use the frequency method for SNP calling, the algorithm determines donors, receivers and fractions of donated depth according to these rules and in this order:

1. Discarded *svars* with distance 0 to a retained *svar* are donors.
2. Donors only donate and never receive depth.
3. Receivers are all *svars* that have distance 0 to a donor.
4. Receivers only receive and never donate depth.
5. The algorithm splits the depth of a donor and donates it proportionally to its receivers' original depths.
6. The original depth of a receiver remains the same until donation from all donors is finished.

These simple and pragmatic rules guarantee that depth never flows across a distance > 0.

Rules 5 and 6 are different if you use the binomial likelihood ratio method for SNP calling (Fig. 5):

5. Donors that have a single receiver donate their complete depth to this receiver.
6. The algorithm discards the depth of other donors.

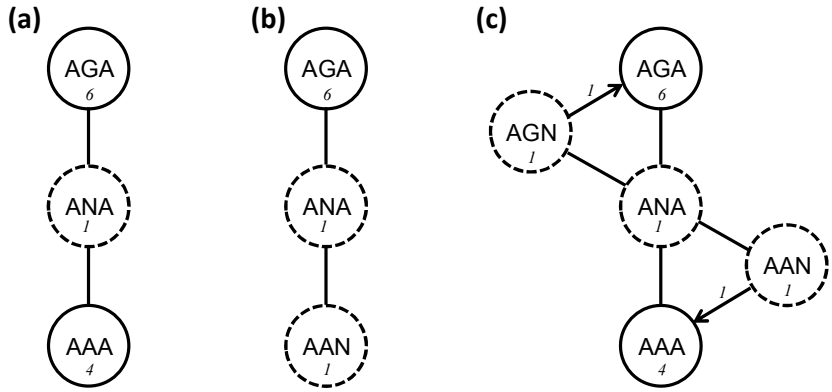

Fig. 5 Treatment of promiscuous networks under the binomial likelihood ratio method for SNP calling. Dashed lines encircle *svars* containing N; solid lines encircle retained *svars*. Numbers in circles depict *svar* depths. Arrows symbolize donation of depth. Numbers at errors depict donated depth.

Loss of alleles

Putative alleles can get lost from a locus in the following steps: During the identification of alleles, the algorithm discards *svars* containing N from promiscuous networks. Such a discarded read variant could represent a distinct allele or alternatively, be a sequence variant of another, retained allele. In such a case, the number of lost putative alleles is unknown. During the filtering of alleles, the algorithm discards alleles containing N. When using the frequency threshold method, the program optionally discards alleles with depths below a threshold. When using the binomial likelihood ratio method, it selects the two alleles with the greatest depths. In these cases, the number o lost putative alleles is known. The program determines the number of lost putative alleles for each individual locus – as far as possible. If it cannot determine the number of lost alleles, it determines the minimum number of lost alleles and stores the information that the true number of lost alleles could be greater. The program prints these data for each locus to outfile *_loci.txt for each individual:

lost_alleles: (minimum) number of lost putative alleles

potlostmore: 1 if the number of lost putative alleles is potentially greater than *lost_alleles*, 0 if not.

Reference

Glaubitz, J.C., Casstevens, T.M., Lu, F., Harriman, J., Elshire, R.J., Sun, Q., Buckler, E.S. (2014) TASSEL-GBS: A high capacity genotyping by sequencing analysis pipeline. *PLoS ONE* 9(2): e90346.