GIbPSs 1.0 Manual Part 9: Documentation of indel_checker_03.0.pl

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

Loci may have alleles that differ by indel variation. Reads of such alleles can have different lengths and must be properly aligned before a correct pairwise distance can be calculated. **indloc** and **poploc** ignore this problem and assign the respective reads to different loci. These loci are in fact pseudo loci, which are indel variants of each other. **indel_checker** identifies loci that could be indel variants of each other. It exports data from the database and uses the program USEARCH (Edgar, R.C. 2010) to analyze them. **indel_checker** analyzes the output from USEARCH and produces outfiles for two other programs: **depth_analyzer** uses the output of **indel_checker** for an improved identification of putative paralogs. **data_selector** enables you to exclude loci that could be indel variants of each other.

Requirements

USEARCH must be installed on your system. The program is available for Linux, Windows and Mac OSX at http://www.drive5.com/usearch/. I have tested indel_checker with USEARCH v7.0.1090.

Before you can use **indel_checker**:

You must run **fdm**, **indloc**, **poploc** and **indpoploc**. You should also run **data_selector** and exclude all loci with a sequence length below 32 if your database contains any. This is a restriction of USEARCH.

Usage

The program understands six command flags. The flags `–UP` and `–DP` are mandatory. The other four flags enable you to make some settings for the analysis with USEARCH. To start the program, enter something like

```
indel_checker_03.0.pl –UP C:\myprograms\usearch\usearch.exe –DP
C:\mydata\main_database –id 0.9 –minsl 0.9 –query_cov 0.9 –target_cov 0.9
```

Command flags (default values in parentheses)

-UP	()	full path to USEARCH
-DP	()	full path to your main database directory
-id	(0.9)	identity threshold
-minsl	(0.9)	minimum value of shorter sequence length / longer sequence length
-query_cov	(0.9)	minimum fraction of query sequence aligned
-target_cov	(0.9)	minimum fraction of target sequence aligned

The flags `–id`, `–minsl`, `–query_cov` and `–target_cov` are explained in detail in the documentation of USEARCH and briefly below under algorithm overview.

Next steps after indel_checker (recommended workflow)

Call **depth_analyzer** to identify repetitive elements. It will use the output of **indel_checker** for an improved analysis. Use **data_selector** to exclude indel variant loci identified by **indel_checker**. Use this option: `Select loci: List of loci in a file: indelcheck/no_indel_loc.txt`. Call **pair_finder** to identify *rc-locus-pairs*. Use **data_selector** to select one locus of each pair. Use **data_selector** to filter loci based on the output of **depth_analyzer** and further criteria.

Algorithm overview

The program loads your current selection of loci if it finds one in directory `export` or loads all loci in the database if not. It selects one allele (popall) of each locus (poploc), sorts the unmerged sequences of these alleles by decreasing length and saves them in a file `popall.fasta` in a new directory `indelcheck` in the main database directory. The program always uses unmerged sequences from all loci, even when **poploc** has merged sequences of loci with a specific length. The rationale behind this is that alleles of a locus vary in length when there is indel variation. Merging requires sufficient overlap of the two fragments from forward and reverse reads. For certain loci, merging will thus be possible for the shorter alleles but not for the longer ones. The detection of pseudo loci, which are indel variants of each other, is thus easier based on unmerged sequences.

Next, **indel_checker** calls USEARCH with the file `popall.fasta` as infile. With the settings described above, it will call USEARCH as follows:

```
C:\myprograms\usearch\usearch.exe –cluster_smallmem
C:\mydata\main_database\indelcheck\popall.fasta –id 0.9 –fulldp –minsl 09 –
query_cov 0.9 –target_cov 0.9 –uc
C:\mydata\main_database\indelcheck\u_out.txt
```

Please refer to the documentation of USEARCH for full details. I give only a brief explanation here. The program uses the UCLUST algorithm of USEARCH with the cluster_smallmem command. This algorithm clusters sequences with a centroid-based method. The first sequence that founds a cluster is the centroid (target). Additional sequences (query) are added to the cluster when they have a similarity to the centroid above the identity threshold specified by –id. The program uses a global alignment to calculate the identity. Alignment heuristics are turned off (`–fulldp`). A threshold of 0.9 means that 90 % of the alignment colums excluding terminal gaps must contain identical letters. USEARCH offers a number of additional optional criteria for the successful assignment of a query sequence to a target (centroid). **indel_checker** uses three of them, `–minsl`, `–query_cov` and `–target_cov`, which are explained above and in the documentation of USEARCH. The flag `–uc` specifies a certain type of outfile, which USEARCH will save in directory `indelcheck` in the main database directory. **indel_checker** analyzes this outfile and produces three further outfiles in directory `indelcheck`.

indel_checker conservatively treats all allele sequences in clusters with more than one sequence as representatives of pseudo loci that are indel variants of each other. In fact, not all clusters with more than one sequence will contain indel variant sequences. This depends largely on your settings: I recommend using an identity threshold that somehow corresponds to your distance settings for **indloc** and **poploc**. You should also consider that USEARCH calculates the percent identity under exclusion of terminal gaps. I recommend using `–minsl` to narrow the sequence length range in a cluster and `–query_cov` and `–target_cov` to ascertain an appropriate overlap of query and target.

Outfiles

All outfiles out of **indel_checker** and USEARCH are in directory `indelcheck` in the main database directory. The program gives an error message and stops execution when this directory already exists.

`popall.fasta`

This file contains the allele sequences in fasta format. The fasta header consist of the poplocID and popallID separated by an underscore.

`u_out.txt`

-uc outfile out of USEARCH. The format is explained in the documentation of USEARCH.

`indelcheck_report`

This file contains information about your selection of data before you started **indel_checker**, files produced by **indel_checker** and USEARCH, how i**ndel_checker** called USEARCH and summary statistics. Example:

```
Summary statistics:
23045 loci
21383 clusters
19961 loci are not indel variants of other loci.
3084 loci could be indel variants of other loci and are clustered into
1422 clusters.
6.65014263667399% of clusters contain more than one locus.
```

`clusters.txt`

This tab-delimited text file with header line contains two columns: Cluster_No: cluster number identified by USEARCH, poplocID: poplocID of a locus in that cluster. **depth_analyzer** automatically uses this file when it finds it to perform a depth analysis based on clusters instead of poplocs.

`no_indel_loc.txt`

This text file contains the poplocIDs of all loci that have not been identified as putative indel variants of another locus, i.e. have been assigned to a cluster with a single sequence by USEARCH. Use this file to select only these loci with **data_selector** as follows: `Select loci: List of loci in a file: indelcheck/no_indel_loc.txt`

Infiles

The program needs the following infiles. It produces an error message and stops execution when it cannot open a file.

Infile	produced by
`export/sel_loc.txt`	data_selector
`poploc.txt`	poploc
`popall.txt`	poploc

Reference

Edgar, R.C.(2010) Search and clustering orders of magnitudes faster than BLAST, *Bioinformatics* 26(19), 2460-2461.