discoSnp

an efficient tool for discovering SNPs without a reference genome

v1.2.2

User's guide – May 2014

contact: pierre.peterlongo@inria.fr

CeCILL License

Publication

Publication is in preparation

IN/OUT in a few words

Software **discoSnp** is designed for extracting Single Nucleotide Polymorphism (SNP) from raw set(s) of reads obtained with Next Generation Sequencers (NGS).

The number of input read sets is not constrained, it can be one, two, or more. No other data such as a reference genome or annotations are needed.

Note that this tool is specially designed to use only a limited amount of memory (3 billions reads of size 100 can be treated with less that 6GB memory).

The software is composed of two modules. The first module, **kissnp2**, detects SNPs from read sets. The second module, **kissreads**, enhances the kissnp2 results by computing per read set and for each found SNP i/ its mean read coverage and ii/ the average (phred) quality of reads generating the polymorphism.

Download

From discoSnp web page – http://colibread.inria.fr/discosnp/ . Please read and accept the CeCILL license before downloading.

Installation

- Unzip the downloaded package :

  # unzip discoSnp_versionnumber.zip
- Get into the newly created *discoSnp* directory:

  # cd discoSnp
- Compile the two modules (kissnp and kissreads) with the single command:

  # ./compile_kissnp_kissreads.sh k_value

  Note about the k value:

  Replace *k_value* by the value of the kmer that you wish to use.
  No need to recompile for any values below or equal to 31
  No need to recompile for any values in [32,63]
  No need to recompile for any values in [64,127]
  ...
  By default, *k*=31 (value used both for compiling and in runtime).

- If you wish to recompile (for instance for changing the *k* value from any value below or equal to 31 to any value bigger or equal to 32) first clear the previously compiled executables:

  # ./clear_kissnp_kissreads.sh

This installation process should generate the “tool” directory containing the two executables : *kissnp2* and *kissreads.*

Running discoSnp

- **Modify the *run_discoSnp.sh* file header or use it with parameters (see below).** You will provide the following information:
  - *read_sets="readref.fasta readsnp.fastq.gz”*: localization of the read files. Note that these files may be in fastq, or fasta, gzipped or not.
  - *prefix="test_ref" :* all temp and final files will be written will start with this prefix
  - *k=31:* size of kmers
  - b=0 branching filtering approach. This parameters influances the precision recall.
    - 0: SNPs for wich any of the two paths is branching are discarted (high precision, lowers the recal in complex genomes). Default value
    - 1: (smart branching) forbid SNPs for wich the two paths are branching (e.g. the two paths can be created either with a 'A' or a 'C' at the same position
    - 2: No limitation on branching (lowers the precision, high recall)"
  - *c=4:* minimal kmer coverage
  - *d=1: max* number of errors per read (used by kissreads only)
  - *g=10000000 :* estimated genome size. Used only to control kissnp2 memory usage. e.g. 3 billion (3000000000) uses 4Gb of RAM.
  - PATH_RS="./tools/": where the two executables are. If you decide to copy them in a directory member of the PATH environment variable (e.g. /usb/local/bin), just set PATH_RS=””
- Additionally you may change some kissnp2 / kissreads options. In this case you may change the two corresponding lines in the *run_discoSnp.sh* file. To know the possible options, type

./tools/kissnp2 and/or *./tools/kissreads* without options. Note that usually, changing these options is not necessary.

- Note that **you may also specify the parameters in command line fashion**. For instance:
  # ./run_discoSnp.sh -r *"data_sample/reads_sequence1.fasta data_sample/reads_sequence2.fasta.gz" -p test_ref -k 31 -c 4 -d 1 -g 10000000 -b 2*
  - Note that command line parameters have the higher priority than those hard coded in the .sh file.
  - Just type "./run_discoSnp.sh -h" to obtain the list of parameters.

- **Sample example**:
  - By default, if you just type:
    # ./run_discoSnp.sh
    discoSnp will be run on a toy example of reads contained in the *data_sample* directory. Among others this will generate the file *test_ref_k_31_c_4_coherent.fa* containing the found SNPs, ranked with respect to their coverages.

Output

- **Final results are in *prefix*_coherent_k_*kval*_c_*cval*.fa file.** This is a simple fasta file composed of a succession of pairs of sequences. Each pair corresponds to a SNP. Let's look at an example :

  >SNP_higher_path_2|high|left_unitig_length_472|right_unitig_length_261|left_contig_length_472|
  right_contig_length_378|C1_8|C2_120|rank_0.88900

  ttgcggataccgttgagacatcttataagtagacgcaatgcggaatcttatagaatcgcccgatagcgttgtgttggtggacacggctgattaccctctcaccc
  gcgctattagcttccataccacctgcggccatccattaagatccgctgctcctcacgaaaaaagaattaataagaagtcccgtaacatgcggatttggtagtc
  gttatagacaactttactggggcgaactaaaacgcttgtggacagaattttggcagtggcaattaatctctaatgatgtgatattagggtctaaaatgtaagaa
  ttcggtgagttagattggacaaggggatccgaagatgttttggcgcagttagtcacagggggagcccctgcctacaaaaagcgcttactgttgactgtctag
  ggatacagcgaaagcggcagtcgttgaagcaaaagtgatatgtgcgacactgcatctagGCAGCGCAACAACGCAACAGCTCGAGG
  *TGT**A**CTTCGCAGAGAAACCGCACGTCCAGTTCTAacactctcatatgtgctcgtcgtttatgctttcggcgtgaaaactggtgcgccg*
  gtgtctggagaccatccttcttgcgtatgactccaaggacagccatcacggtttgtgggttcactgggactgtcacgcttaaccggacggaactcgagaagg
  catacgactggtcgtaagaccgctctgatccgacaccaccataacgcggcactcatgattatcatcactttttttagtccctattacagagctgccgggtggatg
  actctctaccgcgctctgtggaagtgcacttgatcgttttgctgtagaaaaaacttaataaacagaatgccgatgaaggcactactgtactaatagggccggg
  ctacatgttaactac

  >SNP_lower_path_2|high|left_unitig_length_472|right_unitig_length_261|left_contig_length_472|
  right_contig_length_378|C1_118|C2_6|rank_0.88900

  ttgcggataccgttgagacatcttataagtagacgcaatgcggaatcttatagaatcgcccgatagcgttgtgttggtggacacggctgattaccctctcaccc
  gcgctattagcttccataccacctgcggccatccattaagatccgctgctcctcacgaaaaaagaattaataagaagtcccgtaacatgcggatttggtagtc
  gttatagacaactttactggggcgaactaaaacgcttgtggacagaattttggcagtggcaattaatctctaatgatgtgatattagggtctaaaatgtaagaa
  ttcggtgagttagattggacaaggggatccgaagatgttttggcgcagttagtcacagggggagcccctgcctacaaaaagcgcttactgttgactgtctag
  ggatacagcgaaagcggcagtcgttgaagcaaaagtgatatgtgcgacactgcatctagGCAGCGCAACAACGCAACAGCTCGAGG
  *TGT**T**CTTCGCAGAGAAACCGCACGTCCAGTTCTAacactctcatatgtgctcgtcgtttatgctttcggcgtgaaaactggtgcgccgg*
  tgtctggagaccatccttcttgcgtatgactccaaggacagccatcacggtttgtgggttcactgggactgtcacgcttaaccggacggaactcgagaaggc
  atacgactggtcgtaagaccgctctgatccgacaccaccataacgcggcactcatgattatcatcactttttttagtccctattacagagctgccgggtggatga
  ctctctaccgcgctctgtggaagtgcacttgatcgttttgctgtagaaaaaacttaataaacagaatgccgatgaaggcactactgtactaatagggccgggc
  tacatgttaactac

- In this example a SNP A/T is found (underlined here). The central sequence of length 2k-1 (here 2*31-1=61) is seen in upper case, while the two (left and right) extensions are seen in lower case.

- The comments are formatted as follow :

  >SNP_*higher/lower*_path_*id*|*high/low*|*left_unitig_length_int*|*right_unitigtig_length_int*|
  left_contig_length_int|*right_contig_length_int*|C1_*int*|C2_*int*|[Q1_*int*|Q2_*int*|]rank_*float*

  - *higher/lower:* one of the two alleles

- *id:* id of the SNP: each SNP (couple of sequences) has a unique id, here 3.
- *high/low*: sequence complexity. If the sequece if of low complexity (*e.g.* ATATATATATATATAT) this variable would be *low*
- left_unitig_length: size of the full left extension. Here 472
- right_unitig_length: size of the right extension. Here 261
- left_contig_length: size of the full left extension. Here 472
- right_contig_length: size of the right extension. Here 378
- C1: number of reads mapping the central upper case sequence from the first read set
- C2: number of reads mapping the central upper case sequence from the second read set
- C3 [if input data were at least 3 read sets]: number of reads mapping the central upper case sequence from the third read set
- C4, C5, ...
- Q1 [if reads were given in fastq]: average phred quality of the central nucleotide (here A or T) from the mapped reads from the first read set.
- Q2 [if reads were given in fastq]: average phred quality of the central nucleotide (here A or T) from the mapped reads from the second read set.
- Q3 [if the data were at least 3 fastq read sets]: average phred quality of the central nucleotide (here A or T) from the mapped reads from the third read set.
- Q4, Q5, …
- rank: ranks the predictions according to their read coverage in each condition favoring SNPs that are discriminant between conditions (Phi coefficient) (see publication)

Extensions: differences between unitig and contigs (from version 2.1.1.3)

By default in the pipeline, the found SNPs (of length 2k-1) are extended using a contiger. The output contains such contigs and their lengths are shown in the header (left_contig_length and right_contig_length). Moreover, a contig may hide some small polymorphism such as substitutions and/or indels. The output also proposes the length of the longuest extension not containing any such polymorphism. These extensions are called unitigs (defined as « A uniquely assembleable subset of overlapping fragments »).

Output Analyze

- **From a fasta format to a csv format:** If you wish to analyze the results in a tabulated format:
  - *# python output_analyses/discoSnp_to_csv.py discoSnp_output.fa*
  - will output a .csv tabulated file containing on each line the content of 4 lines of the .fa, replacing the '|' character by spaces and removing the CX_
  - example with previously used SNP example:

```
>SNP_higher_path_3 high left_contig_length_86 right_contig_length_52 78 5 rank_0.89839
tctctaccgcgctctgtggaagtgcacttgatcgttttgctgtagaaaaaacttaataaacagaatgccgatgaaggcactactgtACTAATAGGGCCGGGCTACATGTTAACTACAAGGCTA
TAACCTATTGATGACCCGGTCCATacataacttggtatcgtgcatgtagcgttcaagggctatagcaattccgacg >SNP_lower_path_3 high left_contig_length_86
right_contig_length_52 4 91 rank_0.89839
tctctaccgcgctctgtggaagtgcacttgatcgttttgctgtagaaaaaacttaataaacagaatgccgatgaaggcactactgtACTAATAGGGCCGGGCTACATGTTAACTACTAGGCTA
TAACCTATTGATGACCCGGTCCATacataacttggtatcgtgcatgtagcgttcaagggctatagcaattccgacg
```

- **Genotyping the results**: If you wish to genotype your results:
  - *#python output_analyses/discoSnp_to_genotypes.py discoSnp_output.fa threshold_value*
  - will output a file containing on each line the "genotypes" of a SNP. For each input data set it indicates if the SNP is:
    - heterozygous ALT1 path (coverage ALT1 >= threshold and ALT2 < threshold): **1**
    - heterozygous ALT2 path (coverage ALT1 < threshold and ALT2 >= threshold): **-1**
    - homozygous (coverage ALT1 >= threshold and ALT2 >= threshold): **2**
    - absent (coverage ALT1 < threshold and ALT2 < threshold): **0**
  - then it outputs the central sequence of length 2k-1 replacing the central position by ALT1/ALT2
  - example with previously used SNP example and threshold 20:

```
GENOTYPES_SNP_3_THRESHOLD_20 1 -1 TACTAATAGGGCCGGGCTACATGTTAACTACA/TAGGCTATAACCTATTGATGACCCGGTCCATA
```

Misc.

- Running discoSnp generates a set of files used for indexing the reads. This are all files with suffixes:
  .solid_kmers_binary, *.reads_binary*, *.false_positive_kmers*, *.debloom2*, and *.debloom*
  With large read sets, these files may consume disk space. You may remove all of them after the discoSnp pipeline is over:
  *# rm -f *.solid_kmers_binary *.reads_binary *.false_positive_kmers *.debloom2 *.debloom*
  Note that if you wish to re-run the pipeline without changing option among *k*, *c* and *C* (in *kissnp2_exe*) it is smart not to remove these files as they will be recomputed.