discoSnp++

Reference-free detection of SNPs and small indels

v1.0.0

User's guide – January 2015

contact: pierre.peterlongo@inria.fr

Table of contents

GNU AFFERO GENERAL PUBLIC LICENSE

Publication

R. Uricaru, G. Rizk, V. Lacroix, E. Quillery, O. Plantard, R. Chikhi, C. Lemaitre, and P. Peterlongo, "Reference-free detection of isolated SNPs.," Nucleic acids research, vol. 33, pp. 1–11, Dec. 2014.

discoSnp++ features at a glance

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

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

Quick starting

- Download and uncompress the discoSnp++

- Compile programs:
  - "./compile_discoSnp++.sh"
- Run the simple example:
  - ./run_discoSnp++.sh -r "data_sample/reads_sequence1.fasta.gz data_sample/reads_sequence2.fasta.gz"

This creates a fasta file called ***discoRes_k_31_c_4_D_0_P_1_b_0_coherent.fa*** containing the found SNPs.

Download and install

Download from discoSnp++ web page – http://colibread.inria.fr/discosnp/ *. Please read and accept the license before downloading.*

- Uncompress the downloaded package :
  # tar -xvzf DiscoSNP-[Version_number]-Source.tar.gz
- Get into the newly created *discoSnp* directory:
  # cd DiscoSNP-[Version_number]-Source
- Compile all three tools (graph creation, kissnp2, kissreads)
  # ./compile_discoSnp++.sh

Running discoSnp++

- The main script ***run_discoSnp++.sh*** automatically runs the two modules (SNP detection and read coverage and quality computations). You will provide the following information:
  - -r (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. If there are more than one read file, then they must be surrounded by the " character and they are separated by space. This is the only mandatory parameter.
  - -g: reuse a previously created graph (.h5 file) with same prefix and same k, c and C parameters. Using this option enables to reuse a graph created during a previous experiment with same prefix name same k, c and C values. Else, by default, if such a graph exists, it is overwritten. WARNING: use this option only if you are sure the read set(s) used are the same than those previously used for creating the graph.
  - -b branching_strategy: 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)"
  - [beta] -D value. If specified, discoSnp++ will search for deletions of size from 1 to D included. Default=0
  - [beta] -P value. discoSnp++ will search up to P SNPs in a unique bubble. Default=1.
  - -p prefix_name: All out files will start with this prefix. Default="discoRes"
  - -l: accept low complexity bubbles
  - -k kmer_size: size of kmers (default: 31)
  - -t: extends each polymorphism with left and right unitigs
  - -T: extends each polymorphism with left and right contigs
  - -c value. Minimal coverage per read set: Used by kissnp2 (don't use kmers with lower coverage) and kissreads (read coherency threshold). Default=4

    - [beta] -C value. Maximal coverage per read set: Used by kissnp2 (don't use kmers with higher coverage). Default=2^31-1
    - -d error_threshold: max number of errors per read (used by kissreads only). Default 1
    - -h: show help.
- 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, run *.kissnp2* and/or *kissreads* without options. Note that usually, changing these options is not necessary.

- **Sample example**:
    - You can test discoSnp++ on a toy example containing 3 SNPs. In the discoSnp++ directory, type:

./run_discoSnp++.sh -r "data_sample/reads_sequence1.fasta data_sample/reads_sequence2.fasta.gz"

Output

- **Final results are in *discoRes_k_31_c_4_D_0_P_1_b_0_coherent.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_3|high|left_unitig_length_86|right_unitig_length_261|left_contig_length_169|right_contig_length_764|C1_0|C2_134|rank_1.00000

  cgtcggaattgctatagcccttgaacgctacatgcacgataccaagttatgtatggaccgggtcatcaataggttatagcccttgtagttaacatgtagcccggc
  cctattagtacagtagtgccttcatcggcattctgtttattaagttttttctacagcaaaacgatCAAGTGCACTTCCACAGAGCGCGGTAGA
  *GA**G**TCATCCACCCGGCAGCTCTGTAATAGGGACtaaaaaagtgatgataatcatgagtgccgcgttatggtggtgtcggatcagag*
  cggtcttacgaccagtcgtatgccttctcgagttccgtccggttaagcgtgacagtcccagtgaacccacaaaccgtgatggctgtccttggagtcatacgca
  agaaggatggtctccagacaccggcgcaccagttttcacgccgaaagcataaacgacgagcacatatgagagtgttagaactggacgtgcggtttctctg
  cgaagaacacctcgagctgttgcgttgttgcgctgcctagatgcagtgtcgcacatatcacttttgcttcaacgactgccgctttcgctgtatccctagacagtc
  aacagtaagcgctttttgtaggcaggggctccccctgtgactaactgcgccaaaacatcttcggatccccttgtccaatctaactcaccgaattcttacatttta
  gaccctaatatcacatcattagagattaattgccactgccaaaattctgtccacaagcgttttagttcgccccagtaaagttgtctataacgactaccaaatccg
  catgttacgggacttcttattaattcttttttcgtgaggagcagcggatcttaatggatggccgcaggtggtatggaagctaatagcgcgggtgagagggtaat
  cagccgtgtccaccaacacaacgctatcgggcgattctataagattccgcattgcgtctacttataagatgtctcaacggtatccgcaa

  >SNP_lower_path_3|high|left_unitig_length_86|right_unitig_length_261|left_contig_length_169|right_contig_length_764|C1_124|C2_0|rank_1.00000

  cgtcggaattgctatagcccttgaacgctacatgcacgataccaagttatgtatggaccgggtcatcaataggttatagcccttgtagttaacatgtagcccggc
  cctattagtacagtagtgccttcatcggcattctgtttattaagttttttctacagcaaaacgatCAAGTGCACTTCCACAGAGCGCGGTAGA
  *GA**C**TCATCCACCCGGCAGCTCTGTAATAGGGACtaaaaaagtgatgataatcatgagtgccgcgttatggtggtgtcggatcagag*
  cggtcttacgaccagtcgtatgccttctcgagttccgtccggttaagcgtgacagtcccagtgaacccacaaaccgtgatggctgtccttggagtcatacgca
  agaaggatggtctccagacaccggcgcaccagttttcacgccgaaagcataaacgacgagcacatatgagagtgttagaactggacgtgcggtttctctg
  cgaagaacacctcgagctgttgcgttgttgcgctgcctagatgcagtgtcgcacatatcacttttgcttcaacgactgccgctttcgctgtatccctagacagtc
  aacagtaagcgctttttgtaggcaggggctccccctgtgactaactgcgccaaaacatcttcggatccccttgtccaatctaactcaccgaattcttacatttta
  gaccctaatatcacatcattagagattaattgccactgccaaaattctgtccacaagcgttttagttcgccccagtaaagttgtctataacgactaccaaatccg
  catgttacgggacttcttattaattcttttttcgtgaggagcagcggatcttaatggatggccgcaggtggtatggaagctaatagcgcgggtgagagggtaat
  cagccgtgtccaccaacacaacgctatcgggcgattctataagattccgcattgcgtctacttataagatgtctcaacggtatccgcaa

- In this example a SNP G/C 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 86
- right_unitig_length: size of the right extension. Here 261
- left_contig_length: size of the full left extension. Here 169
- right_contig_length: size of the right extension. Here 764
- 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 comma ',' and removing the CX_
  - example with previously used SNP example:

    python output_analyses/discoSnp++_to_csv.py discoRes_..._coherent.fa

    >SNP_higher_path_3,high,left_unitig_length_86,right_unitig_length_261,0,134,rank_1.00000,*[fasta_sequence1]*,>SNP_lower_path_3,high,left_unitig_length_86,right_unitig_length_261,124,0,rank_1.00000,*[fasta_sequence2]*
- **Genotyping the results**: If you wish to genotype your results:

- *#python output_analyses/discoSnp++_to_genotypes.py discoRes_..._coherent.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:

```
python .output_analyses/discoSnp++_to_genotypes.py discoRes_..._coherent.fa 20
GENOTYPES_SNP_3_THRESHOLD_20 -1 1 CAAGTGCACTTCCACAGAGCGCGGTAGAGAG/CTCATCCACCCGGCAGCTCTGTAATAGGGAC
GENOTYPES_SNP_2_THRESHOLD_20 -1 1 ACTAATAGGGCCGGGCTACATGTTAACTACT/AAGGCTATAACCTATTGATGACCCGGTCCAT
GENOTYPES_SNP_1_THRESHOLD_20 1 -1 GCAGCGCAACAACGCAACAGCTCGAGGTGTT/ACTTCGCAGAGAAACCGCACGTCCAGTTCTA
```