discoSnp++

Reference-free detection of SNPs and small indels

v2.2.X

User's guide – April 2016

contact: pierre.peterlongo@inria.fr – r**emarks and questions**: http://www.biostars.org/t/discosnp/

Table of contents

GNU AFFERO GENERAL PUBLIC LICENSE

Publications

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.

C. Riou, C. Lemaitre, and P. Peterlongo, "VCF_creator: Mapping and VCF Creation features in DiscoSnp++". Poster at Jobim 2015

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 three independent modules. The first module, **kissnp2**, detects SNPs and indels from read sets. The second module, **kissreads2**, 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. The third module,

VCF_creator generates a VCF file from the kissnp2/kissreads2 outputs. VCF_creator may use a reference file or not.

Quick starting

- Download and uncompress the discoSnp++ from the discoSnp github page:
  - "https://github.com/GATB/DiscoSnp"
- Compile programs (for source versions):
  - "sh INSTALL"
- Run the simple example:
  - "./run_discoSnp++.sh -r test/fof.txt -T"

This creates a fasta file called ***discoRes_k_31_c_4_D_0_P_1_b_0_coherent.fa*** containing the found SNPs *and a VCF file called* ***discoRes_k_31_c_auto_D_0_P_1_b_0_withlow_coherent.vcf*** *containing the same variants in a VCF fashion.*

Get a local copy of DiscoSnp source code

git clone --recursive https://github.com/GATB/DiscoSnp.git

 # compile the code an run a simple test on your computer

cd DiscoSnp

sh INSTALL

Getting a binary stable release

Binary release for Linux and Mac OSX are provided within the "Releases" tab on Github/DiscoSnp web pag (https://github.com/GATB/DiscoSnp)

After downloading and extracting the content of the binary archive, please run the following command from DiscoSnp home directory:

chmod +x run_discoSnp++.sh test/.sh scripts/*.sh*

Running discoSnp++

The main script ***run_discoSnp++.sh*** automatically runs the three modules (1/ SNP detection; 2/ read coverage and quality computations; 3/ VCF creation). You will provide the following information:

OPTIONS RELATED TO VARIANT CALLING

- -r (read_sets) file_of_files.txt: File indicating the localization of the read files. Note that these files may be in fastq, or fasta, gzipped or not.
  This is the only mandatory parameter.
  See the dedicated section describing the file of file format.
- -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 influences the precision recall.
  - 0: variants for which any of the two paths is branching are discarded (high precision, lowers the recall 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)"
- -s value. In b2 mode only: maximal number of symmetrical croasroads traversed while trying to close a bubble. Default: no limit
- -D value. If specified, discoSnp++ will search for deletions of size from 1 to D included. Default=0
- -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: remove 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). A kmer is conserved only if its coverage is “solid” in a read set. For being “solid” for a read set, a kmer coverage must be higher or equal to the threshold of this read set. Default=auto.
  The minimal coverage may be **either** :
  - Equal to a fixed value, equal for all read sets.
    - Example: -c 4
  - Distinct for each read set
    - Example with three read sets: -c 4,5,17
  - Automatically detected
    - Example -c auto
  - Automatically detected for one or some of the read sets:
    - Example with three read sets: -c 4,auto,7

  Note the if the number of provided values is lower than the number of read sets, the last value applies for all remaining read sets:
    - Example with three read sets:
      - -c 4,7 applies -c 4,7,7
      - -c 4,auto applies -c 4,auto,auto
- -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 **kissreads2** only). Default 1
- -n: do not compute the genotypes
- -u: max number of used threads

Options related to VCF_Creator AND to the possible usage of a reference genome

- -G reference_file: reference genome file (fasta, fastq, gzipped or nor). In absence of this file the VCF created by VCF_creator won't contain mapping related results
- -R: use the reference file also in the variant calling, not only for mapping results. In this case, all the kmers of the reference file are used for calling variants. Note that the kissreads2 results won't show read coverage for the reference.
- -B: bwa path . e.g. /home/me/my_programs/bwa-0.7.12/ (note that bwa must be pre-

compiled)
        - Optional unless option -G used and bwa is not in the binary path
    - -M: Maximal number of mapping errors during BWA mapping phase. Default 4
        - Useless unless mapping on reference genome is required (option -G).

Additionally you may change some kissnp2 / kissreads2 / VCF_creator options. In this case you may change the two corresponding lines in the *run_discoSnp++.sh* file. To know the possible options, run *./build/tools/kissnp2/kissnp2* and/or *./build/tools/kissreads2/kissreads2* and/or *./run_VCF_creator.sh* without options. Note that usually, changing these options is not necessary.

Input file of files format:

The input read sets are provided using a file of file(s). The file of file(s) contains on each line a read file or another file of file(s).

Let's look to a few usual cases (italic strings indicate the composition of a file):

- Case1: I've a unique read set composed of a unique read file (reads.fq.gz).
    - fof.txt:
        reads.fq.gz
- Case2: I've a unique read set composed of a couple of read files (reads_R1.fq.gz and reads_R2.fq.gz). This may be the case in case of pair end sequencing.
    - fof.txt:
        fof_reads.txt:
        - with fof_reads.txt:
            reads_R1.fq.gz
            reads_R2.fq.gz
- Case3: I've two read sets each composed of a unique read file: reads1.fq.gz and reads2.fq.gz:
    - fof.txt:
        reads1.fq.gz
        reads2.fq.gz
- Case4: I've two read sets each composed two read files: reads1_R1.fq.gz and reads1_R2.fq.gz and reads2_R1.fq.gz and reads2_R2.fq.gz:
    - fof.txt:
        fof_reads1.txt
        fof_reads2.txt
        - fof_reads1.txt
            reads1_R1.fq.gz
            reads1_R2.fq.gz
        - fof_reads2.txt:
            reads2_R1.fq.gz
            reads2_R2.fq.gz
- *and so on...*

Correspondance between read set file names and the Ci or Gi values

DiscoSnp++ provides a file ${prefix}_read_files_correspondance.txt (with ${prefix} provided by the -p option, equal to “discoRes” by default) provinding the correspondance between the read file names, and the read sets. Given the previously provided examples, this file would contain:

- case1:

  C_1 *reads.fq.gz*

- case2:

C_1 *reads_R1.fq.gz reads_R2.fq.gz*

- case3:

  C_1 *reads_R1.fq.gz*

  C_2 *reads_R2.fq.gz*

- case4:

  C_1 *reads1_R1.fq.gz reads1_R2.fq.gz*

  C_2 *reads2_R2.fq.gz reads2_R2.fq.gz*

Sample example:

You can test discoSnp++ on a toy example containing 3 SNPs. In the discoSnp++ directory, type: *./run_discoSnp++.sh -r fof.txt -T*

(use -T in order to obtain the left and right contigs of each found polymorphism)

Output

(Results with close SNPs and indels are given at the end of this document)

- **Final fasta results are in *discoRes_k_31_c_auto_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|P_1:30_C/G|high|nb_pol_1|left_unitig_length_86|right_unitig_length_261|
  left_contig_length_168|right_contig_length_764|C1_124|C2_0|G1_0/0|G2_1/1|rank_1.00000

  cgtcggaattgctatagcccttgaacgctacatgcacgataccaagttatgtatggaccgggtcatcaataggttatagcccttgtagttaacatgtagcccggc
  cctattagtacagtagtgccttcatcggcattctgtttattaagttttttctacagcaaaacgatCAAGTGCACTTCCACAGAGCGCGGTAGA
  *GA**G**TCATCCACCCGGCAGCTCTGTAATAGGGACtaaaaaagtgatgataatcatgagtgccgcgttatggtggtgtcggatcagag*
  cggtcttacgaccagtcgtatgccttctcgagttccgtccggttaagcgtgacagtcccagtgaacccacaaaccgtgatggctgtccttggagtcatacgca
  agaaggatggtctccagacaccggcgcaccagttttcacgccgaaagcataaacgacgagcacatatgagagtgttagaactggacgtgcggtttctctg
  cgaagaacacctcgagctgttgcgttgttgcgctgcctagatgcagtgtcgcacatatcactttgcttcaacgactgccgctttcgctgtatccctagacagtc
  aacagtaagcgctttttgtaggcaggggctccccctgtgactaactgcgccaaaacatcttcggatcccccttgtccaatctaactcaccgaattcttacatttta
  gaccctaatatcacatcattagagattaattgccactgccaaaattctgtccacaagcgttttagttcgccccagtaaagttgtctataacgactaccaaatccg
  catgttacgggacttcttattaattcttttttcgtgaggagcagcggatcttaatggatggccgcaggtggtatggaagctaatagcgcgggtgagagggtaat
  cagccgtgtccaccaacacaacgctatcgggcgattctataagattccgcattgcgtctacttataagatgtctcaacggtatccgcaa

  >SNP_lower_path_3|P_1:30_C/G|high|nb_pol_1|left_unitig_length_86|right_unitig_length_261|
  left_contig_length_168|right_contig_length_764|C1_0|C2_134|G1_0/0|G2_1/1|rank_1.00000

  cgtcggaattgctatagcccttgaacgctacatgcacgataccaagttatgtatggaccgggtcatcaataggttatagcccttgtagttaacatgtagcccggc
  cctattagtacagtagtgccttcatcggcattctgtttattaagttttttctacagcaaaacgatCAAGTGCACTTCCACAGAGCGCGGTAGA
  *GA**C**TCATCCACCCGGCAGCTCTGTAATAGGGACtaaaaaagtgatgataatcatgagtgccgcgttatggtggtgtcggatcagag*
  cggtcttacgaccagtcgtatgccttctcgagttccgtccggttaagcgtgacagtcccagtgaacccacaaaccgtgatggctgtccttggagtcatacgca
  agaaggatggtctccagacaccggcgcaccagttttcacgccgaaagcataaacgacgagcacatatgagagtgttagaactggacgtgcggtttctctg
  cgaagaacacctcgagctgttgcgttgttgcgctgcctagatgcagtgtcgcacatatcactttgcttcaacgactgccgctttcgctgtatccctagacagtc
  aacagtaagcgctttttgtaggcaggggctccccctgtgactaactgcgccaaaacatcttcggatcccccttgtccaatctaactcaccgaattcttacatttta
  gaccctaatatcacatcattagagattaattgccactgccaaaattctgtccacaagcgttttagttcgccccagtaaagttgtctataacgactaccaaatccg
  catgttacgggacttcttattaattcttttttcgtgaggagcagcggatcttaatggatggccgcaggtggtatggaagctaatagcgcgggtgagagggtaat
  cagccgtgtccaccaacacaacgctatcgggcgattctataagattccgcattgcgtctacttataagatgtctcaacggtatccgcaa

- In this example a SNP G/C is found (underlined here and indicated in the comment). 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*|P_i:pos_Alt1/Alt2|*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.
- **[FOR SNPs]** *P_i:pos_Alt1/Alt2:* Information about a i^{th} SNP (If more than a unique SNP is found, the following format is used: *P_1:pos_Alt1/Alt2,P_2:pos_Alt1/Alt2,...*
  - *pos*: position of the SNP with respect to the starting position of the bubble, i.e. the starting of the upper case sequence.
  - *Alt1*: One of the two alleles
  - *Alt2*: the other

- **[FOR INDELS]** *P_1:pos_size_repeatSize*
  - *pos:* predicted position of the indel with respect to the starting position of the bubble, i.e. the starting of the upper case sequence.
  - *size:* predicted size of the indel
  - *repeatSize:* Size of the longest sequence both prefix of the indel and prefix of the sequence located just after the insertion. **Remark**. This information is useful as the real indel may be located in *[pos, pos+repeatSize]*.
- *high/low*: sequence complexity. If the sequece if of low complexity (*e.g.* ATATATATATATATAT) this variable would be *low*
- *nb_pol:* number of polymorphism.
- 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. Note that (see bellow), the correspondance between Ci and the read file names is provided in file "*discoRes_read_files_correspondance.txt*".
- 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, …
- G1: Genotype of the variant in the first read set (considering the higher path as the reference)
- G2: Genotype of the variant in the second read set (considering the higher path as the
- G3, G4, ...
- 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

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 »).

Second created file (from VCF_Creator):

The previous example: *./run_discoSnp++.sh -r fof.txt -T* also generated a vcf file: *discoRes_k_31_c_auto_D_0_P_1_b_0_coherent.vcf*. As we didn't provided a reference file, the VCF contains no mapping position on a reference. Instead, each variant position correspond to the mapping of itself on its own sequence. For instance:

```
SNP_higher_path_3	196	3	C	G	.	.	Ty=SNP;Rk=1;UL=86;UR=261;CL=166;CR=761	GT:DP:PL:AD
0/0:124:10,378,2484:124,0	1/1:134:2684,408,10:0,134
```

corresponds to the previously explained SNP. It maps at position 196.

By adding a reference file: *./run_discoSnp++.sh -r fof.txt -T -G data_sample/reference_genome.fa* the vcf file includes mapping information:

```
chromosome	117	3	C	G	.	PASS	Ty=SNP;Rk=1;DT=-1;UL=86;UR=261;CL=166;CR=761;Genome=C;Sd=1	GT:DP:PL:AD
0/0:124:10,378,2484:124,0	1/1:134:2684,408,10:0,134
```

See documentation specific to VCF_creator for more information: *doc/vcf_creator_user_guide.pdf*

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_

Exemples of close SNPs and indels

Exemple of a multiple SNP:

```
>SNP_higher_path_766|P_1:30_A/T,P_2:34_C/G|high|nb_pol_2|C1_0|C2_0|C3_28|G1_1/1|G2_1/1|G3_0/0|rank_1.00000
AGCGCACAAGGCGTTAGGCGGGCTGGATATAATGCCGCTGGTCGCCGGGAAACAGGTTGCCATTC
```

>SNP_lower_path_766|P_1:30_A/T,P_2:34_C/G|high|nb_pol_2|C1_45|C2_43|C3_0|G1_1/1|G2_1/1|G3_0/0|rank_1.00000

AGCGCACAAGGCGTTAGGCGGGCTGGATATTATGGCGCTGGTCGCCGGGAAACAGGTTGCCATTC

Note that a unique genotype is proposed for close SNPs

Exemple of an indel:

>INDEL_higher_path_3756|P_1:30_8_3|high|nb_pol_1|C1_28|C2_0|C3_0|G1_0/0|G2_1/1|G3_1/1|rank_1.00000

AGGCGACCGAGAAAATGGAGAACGTGCGCATCGCTGTTTATTAATGCCCGTTCGGCG

>INDEL_lower_path_3756|P_1:30_8_3|high|nb_pol_1|C1_0|C2_42|C3_44|G1_0/0|G2_1/1|G3_1/1|rank_1.00000

AGGCGACCGAGAAAATGGAGAACGTGCGCAAGCGGGCATCGCTGTTTATTAATGCCCGTTCGGCG