<div align="center">

`KisSplice`

De-novo calling alternative splicing events
from RNA-seq data

User's guide, version 1.8.3

August 12, 2013

</div>

# Contents

# 1 KisSplice, at a glance

KisSplice is dedicated to *de-novo* calling of alternative splicing events from one or several RNA-seq datasets. In addition to splicing events, KisSplice detects small indels (<3 nucleotides), and provides a list of potential inexact tandem repeats and SNPs.

# 2 KisSplice software package

## 2.1 CeCILL licence

This software is governed by the CeCILL licence. Details are mentionned in the COPYING file.

## 2.2 Reference

If you use KisSplice in a published work, please cite the following reference:

Gustavo AT Sacomoto, Janice Kielbassa, Rayan Chikhi, Raluca Uricaru, Pavlos Antoniou, Marie-France Sagot, Pierre Peterlongo* and Vincent Lacroix*, *KISSPLICE: de-novo calling alternative splicing events from RNA-seq data*, **BMC Bioinformatics 2012**, 13(Suppl 6):S5

## 2.3 Mailing list

To be informed about new releases, bugs or updates, please subscribe to the mailing list kissplice-users@lists.gforge.inria.fr
To do so, please subscribe at : http://lists.gforge.inria.fr/cgi-bin/mailman/listinfo/kissplice-users Do not hesitate to write to the list for any comment or questions on KisSplice.

## 2.4 Requirements

KisSplice uses the CMake building tool. If it is not directly available on your system, you can download it from http://www.cmake.org/cmake/resources/software.html. KisSplice relies on zlib that is included in most systems, however it is also available from http://zlib.net.

## 2.5 Installation

KisSplice is written in C/C++ and is running on Mac OS X and Linux platforms. A version of python 2.7 is needed.
In order to install KisSplice, you need to:

1. Uncompress the archive file:

```
tar zxvf kissplice-1.x.x.tar.gz
```

2. Go into the corresponding directory:

```
cd kissplice-1.x.x
```

3. Launch cmake:

```
cmake .
```

4. Compile the package:

```
make
```

5. Install the package on your system:

```
sudo make install
```

If you want to install the package in a particular directory /path_to_install/, then replace the step 3 by:

```
cmake . -DCMAKE_INSTALL_PREFIX=/path_to_install/
```

By default, the maximum k-mer size in `KisSplice` is 31. In 64-bits machines it is possible to increase this value to 63. To achieve this change, replace step 3 by:

```
cmake . -DKMERS_OVER_32=ON
```

If you do not install the software, the program is available in the repertory **bin**.

# 3   Usage

## 3.1   Testing `KisSplice` in a couple of minutes

The `sample_example` directory contains two files containing simulated reads from two transcripts of D. melanogaster; `reads1.fa` and `reads2.fa`. There is a single splicing event (exon skipping) between both transcripts and a SNP/sequencing error.

Once you installed `KisSplice`, you can run it with defaults parameters ($k = 25$) on the two fasta files present in the `sample_example` directory of the release:

```
bin/kissplice -r sample_example/reads1.fa -r sample_example/reads2.fa
less results/results_reads1_reads2_k25_coherents_type_0.fa
```

The result files can be found in the results directory:
`results_reads1_reads2_k25_coherents_type_0.fa` contains the SNP/sequencing error and `results_reads1_reads2_k25_coherents_type_1.fa` contains the exon skipping event . The other files are empty. The splicing event is condition-specific with the inclusion isoform covered by 52 reads in condition 2 and the exclusion isoform covered by 9 reads in condition 1. You may want to map the sequences against a reference genome if you have one to see what the output corresponds to. This can be done using blat, GMAP or any other RNA/DNA alignment software.

## 3.2 Options

Type `kissplice -h` to see the options and parameters:

```
kissplice [-h] [-r READFILES] [-k KVAL] [-l LLMAX] [-m LL_MIN]
              [-M UL_MAX] [-g GRAPH_PREFIX] [-o OUT_DIR] [-d PATH_TO_TMP]
              [-t NBPROCS] [-s] [-v] [-u] [-c MIN_COV]
              [-C MIN_RELATIVE_COV] [-z GENOME_SIZE] [-e MIN_EDIT_DIST]
              [-y MAX_CYCLES] [--timeout TIMEOUT] [--version]

kisSplice - splicing event caller

optional arguments:
  -h, --help            show this help message and exit
  -r READFILES          input fasta/q read files (mutiple, such as "-r file1 -r
                        file2...")
  -k KVAL               k-mer size (default=25)
  -l LLMAX              maximal length of the shorter path (default: 2k-1)
  -m LL_MIN             minimum length of the shorter path (default 2k-8)
  -M UL_MAX             maximum length of the longest path (default 1000),
                        skipped exons longer than UL_MAX are not reported
  -g GRAPH_PREFIX       path and prefix to pre-built de Bruijn graph (suffixed
                        by .edges/.nodes) if jointly used with -r, graph used
                        to find bubbles and reads used for quantification
  -o OUT_DIR            path to store the results (default = ./results)
  -d PATH_TO_TMP        specific directory (absolute path) where to build
                        temporary files (default temporary directory otherwise)
  -t NBPROCS            number of cores (must be <= number of physical cores)
  -s                    Will not ouput SNPs (save time)
  -v                    Verbose mode
  -u                    keep the nodes/edges file for unfinished bccs
  -c MIN_COV            an integer, k-mers present strictly less than this
                        number of times in the dataset will be discarded
                        (default 2)
  -C MIN_RELATIVE_COV   a percentage from [0,1), edges with relative coverage
                        below this number are removed (default 0.02)
  -z GENOME_SIZE        estimated genome/transcriptome size (default =
                        1000000000)
  -e MIN_EDIT_DIST      edit distance threshold, if the two sequences (paths)
                        of a bubble have edit distance smaller than this
                        threshold, the bubble is classified as an inexact
                        repeat (default 3)
  -y MAX_CYCLES         maximal number of bubbles enumeration in each bcc. If
                        exceeded, no bubble is output for the bcc (default
                        10000)
  --timeout TIMEOUT     max amount of time (in seconds) spent for enumerating
                        bubbles in each bcc. If not all bubbles are enumerated
                        after this time, no bubble is output for the bcc.
```

```
                          Default : 900
  --version               show program's version number and exit
```

## 3.3    Relative coverage

The option $-C$ edits the De-Bruijn graph constructed by removing edges non covered. It is a local filter. For a specific node, if one of its outgoing edges is covered less than $C$ (in percentage) it is removed. For a small value of $C$, this option allows to remove artificial edges due to overlapping of two k-mers (not supported per the reads) or sequencing errors.

## 3.4    Genome-size option

The $-z$ option is used for a rough estimation of the graph size. It is an estimation of unique $k$-mers contained into the dataset. A good evaluation of this parameter leads to an optimal graph construction step in term of memory consumption and time. On one hand, if the value is too low, the construction will take fewer memory but will take more time. On the other hand, if the value is too high, it will use a lot of memory. By default $z$ is set to a billion (1 000 000 000) which is a good trade-off for most datasets.

## 3.5    Paired-end reads

For now, paired-end reads are not supported. We recommend that you use them as two input files :

```
kissplice -r condition1/1 -r condition1/2
```

# 4    I/O

## 4.1    Input

`KisSplice` may be used either directly from one or several sets of reads, or from a bi-directed de-Bruijn graph.
In the first case, one or several fasta/fastq files containing the reads have to be provided. In the second case, the de-Bruijn graph alone has to be provided (in dot format). In the input fasta/q files, each read should be written on exactly two lines (one for the fasta identifier and one for the sequence).
The input files should have one of the following extensions: `fq, fastq, txt` or `fasta, fa`. `KisSplice` also handles compressed reads (`.fa.gz`)
    To have an example of the input files, run `KisSplice` on the test data (provided in the directory `sample_example`), consisting of two read files in fasta format. You will find the constructed de-Bruijn graph built by `KisSplice` in the `KisSplice` directory.

## 4.2   Output

If it was not provided, a de-Bruijn graph is built for the $N$ fasta files (read) given as input; the following three files are created in the results directory (for $k = XX$):

- `graph_NameReadFile1_NameReadFile2_..._NameReadFileN_kXX.nodes` for the nodes of the de-Bruijn graph

- `graph_NameReadFile1_NameReadFile2_..._NameReadFileN_kXX.edges` for its edges.

- `graph_NameReadFile1_NameReadFile2_..._NameReadFileN_kXX.solid_kmers_binary_with_cou` a binary file relevant for the option `-C`

- `graph_NameReadFile1_NameReadFile2_..._NameReadFileN_kXX.counts` for the k-mers count, created only if `KisSplice` was run with option -C.

In the node file, the 1st column is the node ID, the 2nd its (forward) sequence.

In the file describing the edges, the 1st and 2nd column are the IDs of the nodes that are connected by an edge, the 3rd column codes for the direction of the edge (FF = from the forward sequence of a node to the forward sequence of the other node, RR= from reverse to reverse, FR and RF). Note that if node A is connected with node B by an edge with direction "FF" ("FR"), node B is connected to node A by an edge directed "RR" ("RF") and vice versa.

Moreover, five fasta files for the results are created:

1. `results_NameReadFile1_NameReadFile2_..._NameReadFileN_kXX_type_0.fa`: SNPs or sequencing errors

2. `results_NameReadFile1_NameReadFile2_..._NameReadFileN_kXX_type_1.fa`: alternative splicing events

3. `results_NameReadFile1_NameReadFile2_..._NameReadFileN_kXX_type_2.fa`: inexact tandem repeats

4. `results_NameReadFile1_NameReadFile2_..._NameReadFileN_kXX_type_3.fa`: short indels

5. `results_NameReadFile1_..._NameReadFileN_kXX_type_4 .fa`: all others, composed by a shorter path of length $> 2k - 1$ not being a SNP

If the read files (option -r) are provided as input files, the first four fasta files are checked for read coherency, which means that each nucleotide of each sequence has to be covered by at least one read. If `KisSplice` is used with option -g (only the de-bruijn graph is provided), a check for read coherency is not possible due to missing information about read coverage.

Fasta files are organized as follows, each 4-lines groups are an event

```
> identifier_upper_ path
sequence upper path
> identifier_lower_path
sequence lower path
```

The identifier for the upper path is formatted as follows :

```
>bcc_BB|Cycle_YY|Type_ZZ|upper_path_Length_UU|C1_cov1|C2_cov2[...]|CN_covN|rank_RR
```

- `bcc_BB` : Bi-connected component BB the event belongs to

- `Cycle_YY` : since in each bcc, several cycles may exist, this attribute indicates the ID of the cycle (here: cycle YY) that generated the bubble

- `Type_ZZ`: with ZZ=0,1,2,3, the type also corresponds to the type given by the file name

- `upper_path_Length_UU` : length (in nucleotides) of the sequence of the upper path of the bubble

- `Cn`: with n=1,..., N: coverage of the path using reads from the read file n; coverage is the raw count of reads mapping the path with at least $k$ (i.e. the value specified for the $k-mer$ length) nucleotides.

- `rank_RR` : The rank is close to 1 if the alleles of the event are condition-specific. Otherwise, it is close to 0. Formally, the rank corresponds to the square of the Phi coefficient. In the case where there is one condition, the rank is always 0. In the case where there are more than two conditions, it is calculated for all pairs of conditions, and the maximal phi value is output. Note that the rank is just an indication of the strength of the association between the allele and a condition but this is not a statistical test. In particular, for low counts, the rank may be high but a statistical test may give a negative answer to the question "is there an association between alleles and conditions ?". On the other hand, for very large counts, a statistical test may give a positive answer while the effect is very small. The rank helps to measure the strength of the effect.

The identifier for the lower path provides virtually the same information, but concerning the lower (shorter) path of the bubble. In order to facilitate the further post-processing of these files, the bcc, cycle, type and rank are repeated, although this information is redundant.

If `KisSplice` is run with option -r (read files), the events in the output files are sorted with respect to their rank (this is not possible with option -g due to missing information about read coverage).

## 4.3   Uncoherent Bubbles

In order for a bubble to be considered as coherent, each nt has to be covered by at least one read. Uncoherent bubbles are output in a separate file. In principle, uncoherent bubbles should correspond to artefacts of the DBG (we lose information when we move from reads to k-mers). In practice, real events which have a low coverage may produce uncoherent bubbles. Hence, it may be worth to mine this file if you are interested in unfrequent events.

## 4.4   Unfinished BCCs

For some datasets, enumerating all bubbles is very long. Hence, we set a maximum amount of time (900s by default) that the algorithm spends in each BCC (biconnected component). If after this time, the BCC is not finished, the algorithm stops and moves to the next BCC. In practice, these unfinished BCCs correspond to very repetitive regions of the dataset (transposable elements...). Running KisSplice with -u option enables to output these unfinished BCCs for further inspection (for instance using Cytoscape).

# 5   TroubleShooting

Before running KisSplice on a large dataset, (10 conditions with 100M reads each), which takes time and memory, we advise to first run KisSplice on a subset of your dataset (2 conditions with 10M reads each) and get familiar with the output.

If you encounter problems running KisSplice, this may be due to memory issues. Please try setting your stack size as unlimited:

```
> ulimit -s unlimited
```

If the problem persists, please do not hesitate to contact us.