The 'kissDE' package

Clara Benoit-Pilven [1], ***Camille Marchet*** [2], ***Janice Kielbassa*** [3], ***Audric Cologne*** [1], ***Aurélie Siberchicot*** [1], ***and Vincent Lacroix*** *[1]

[1]Université de Lyon, Université Lyon 1, CNRS UMR5558, Laboratoire de Biométrie et Biologie Evolutive, Villeurbanne, France
[2]Univ Rennes, Inria, CNRS, IRISA, France
[3]Synergie Lyon Cancer, Université Lyon 1, Centre Léon Berard, Lyon, France

*vincent.lacroix@univ-lyon1.fr

April 28, 2020

Abstract

kissDE is a package dedicated to the analysis of count data obtained from the quantification of pairs of variants in RNA-Seq data.
It can be used to study splice variants, where the two variants of the pair differ by the inclusion/exclusion of an exonic or intronic region. It can also be used to study genomic variants (whenever they are transcribed), which differ by a single nucleotide variation (SNV) or an indel.
The statistical framework is based on similar hypotheses as *DESeq2* [1] and includes its normalization method using geometric means. Counts are modelled using the negative binomial distribution. We use the framework of the generalised linear model, and we test for association of a variant with a condition using a likelihood ratio test.
This vignette explains how to use this package.
The workflow for SNPs/SNVs is fully described in Lopez-Maestre et al. [2], the workflow for splicing is fully described in Benoit-Pilven et al. [3]

Package

kissDE 1.8.0

Contents

1 Prerequisites

1.1 Use case

kissDE is meant to work on pairs of variants that have been quantified across different conditions. It can deal with single nucleotide variations (SNPs, mutations, RNA editing), indels or alternative splicing.
As *kissDE* was first designed to be a brick of the *KisSplice* [4] pipeline (web page: http://kissplice.prabi.fr/), the `kissplice2counts` function can be directly applied to the output files from *KisSplice* or *KisSplice2refgenome*. Yet, *kissDE* can also run with any other software which produces count data as long as this data is properly formatted.
kissDE was designed to work with at least two replicates for each condition, which means that the minimal input contains the read counts of the variants for 4 different samples, each couple representing a biological condition and its 2 replicates. There can be more replicates and more conditions, but it is not mandatory to have an equal number of replicates in each condition.

1.2 Install and load *kissDE*

In a *R* session, the *BiocManager* package has first to be installed.

```
install.packages("BiocManager")
```

Then, the *kissDE* package can be installed from *Bioconductor* and finally loaded.

```
BiocManager::install("kissDE")
```

```
library(kissDE)
```

1.3 Quick start

Here we present the basic *R* commands for an analysis with *kissDE*. These commands require an external output file of *KisSplice* , for example '`output_kissplice.fa`' (which is not included in this package). To deal with other types of input files, please refer to section 2.1. The funtions used in *kissDE* are `kissplice2counts`, `qualityControl`, `diffExpressedVariants` and `writeOutputKissDE`. For each function, default values of the parameters are used. For more details on functions and their parameters see section 2. Here we assume that there are two conditions ($condition\_1$ and $condition\_2$) with two biological replicates and we also assume that the RNA-Seq libraries are single-end.

```
counts <- kissplice2counts("output_kissplice.fa")
conditions <- c(rep("condition_1", 2), rep("condition_2", 2))
qualityControl(counts, conditions)
results <- diffExpressedVariants(counts, conditions)
writeOutputKissDE(results, output = "kissDE_output.tab")
```

Note that the functions `kissplice2counts` and `diffExpressedVariants` may take some time to run (see section 4.3 for more details on running time).

2 *kissDE*'s workflow

In this section, the successive steps and functions of a differential analysis with *kissDE* are described.

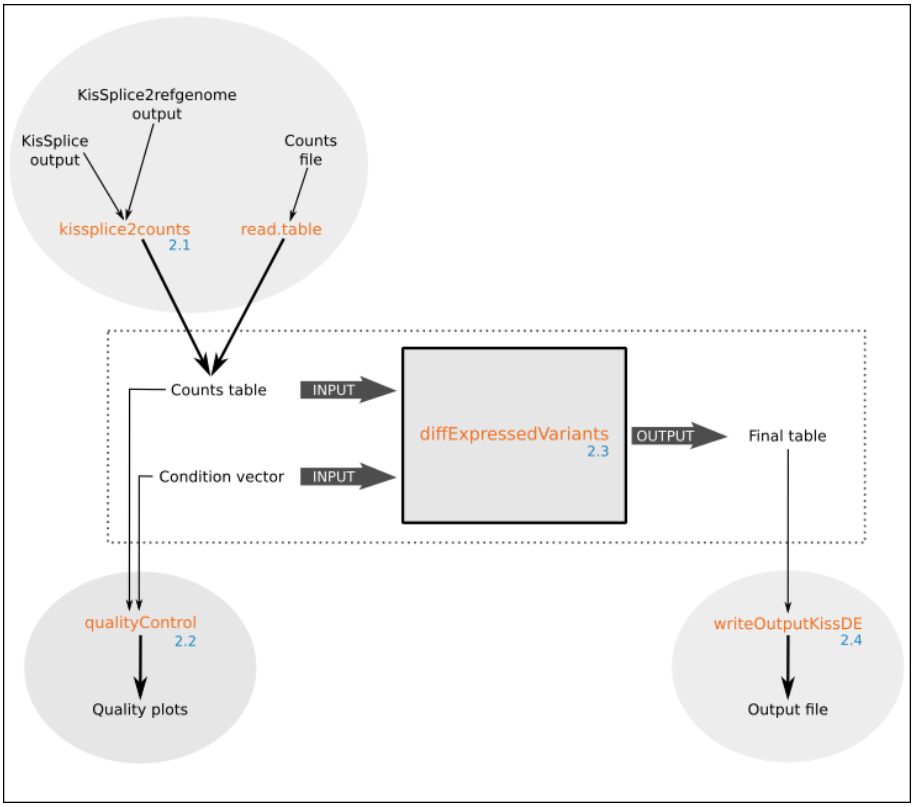

Figure 1: Schema of *kissDE*'s workflow
Numbers in light blue point to the section of this vignette explaining the step.

2.1 Input data

kissDE's input is a table of raw counts and a vector describing the number of conditions and replicates per condition. The table of raw counts can either be directly provided by the user or obtained with *KisSplice* or *KisSplice2refgenome* (http://kissplice.prabi.fr/training/).

2.1.1 Condition vector

The condition vector describes the order of the columns in the count table.
As an example, the counts are ordered as follow: the two first counts represent the two replicates of $condition\_1$ and the two following counts the two replicates of $condition\_2$. In this case, the condition vector for these 2 conditions with 2 replicates per condition, would be:

```
myConditions <- c(rep("condition_1", 2), rep("condition_2", 2))
```

In the case where the input data contains more than 2 conditions, we advise the user to remove samples from the analysis in order to compare 2 conditions only, because *kissDE* was uniquely tested in this context. To remove samples from the analysis the "*" character can be used:

```
myConditions <- c(rep("condition_1", 2), rep("*", 2), rep("condition_3",
    2))
```

Here, there are 3 conditions and 2 replicates per condition, but only $condition\_1$ and $condition\_3$ will be considered in the analysis.
If the count table was loaded from *KisSplice* or *KisSplice2refgenome* output, the condition vector must contain the samples in the same order they were given to *KisSplice* (see sections 2.1.3 and 2.1.4).

Warning: To run kissDE, all conditions must have replicates. So each condition must at least be present twice in the condition vector. If this is not the case, an error message will be printed.

2.1.2 User's own data (without *KisSplice*): table of counts format

Let's assume we work with two conditions ($condition\_1$ and $condition\_2$) and two replicates per condition. An input example table contained in a flat file called 'table_counts_alt_splicing.txt' is loaded and stored in a `tableCounts` object.
Comment: `fpath1` contains the absolute path of the file on the user's hard disk.

```
fpath1 <- system.file("extdata", "table_counts_alt_splicing.txt",
    package = "kissDE")
tableCounts <- read.table(fpath1, head = TRUE)
```

In *kissDE*, the table of counts must be formatted as follows:

```
head(tableCounts)
```

	eventsName	eventsLength	cond1rep1	cond1rep2	cond2rep1	cond2rep2
1	event1	261	105	41	15	26
2	event1	81	2	5	100	150
3	event2	207	20	17	60	58
4	event2	80	58	33	7	1
5	event3	268	53	26	19	29
6	event3	82	3	1	31	55

It must be a data frame with:

- **in rows:**
  - One variation is represented by two lines, one for each variant. For instance, for SNVs, one allele is described in the first line, and the other in the second line. For alternative splicing events, the inclusion isoform and the exclusion isoform have one line each.
  - The header must contain the column names in the flat file.
- **in columns:**
  - The first column (`eventsName`) contains the name of the variation.

- The second column (`eventsLength`) contains the effective size of the variant in nucleotides (bp). The effective size corresponds to the number of read mapping positions used when estimating the abundance of a variant.
  For the exclusion variant (2nd line), which should correspond to an exon-exon junction, it corresponds to:

$$effectiveLengthExclu = readLength - 2 * overhang + 1 \quad (1)$$

  where $overhang$ corresponds to the minimal number of bases needed to accept that a read is aligned to a junction.
  For the inclusion variant (1st line), it corresponds to:

$$effectiveLengthInclu = effectiveLengthExclu + variablePartLength \quad (2)$$

  where $variablePartLength$ is the length of the region only present in the inclusion variant.
  In the special case where the abundance of the inclusion variant has been estimated using only junction reads, then the effective length of the inclusion variant is:

$$effectiveLengthInclu = 2 * effectiveLengthExclu \quad (3)$$

  This information is used only in the context of alternative splicing. In the context of SNVs, it can be set to 0. It is used to assess which splice variants may induce a frameshift (the difference of length between the inclusion and exclusion variant is not a multiple of 3). It is also used to precisely estimate the PSI (Percent Spliced In).
- All other columns (`cond1rep1`, `cond1rep2`, `cond2rep1`, `cond2rep2`) contain read counts of a variant in a sample. In the example above, `cond1rep1` is the number of reads supporting this variant in the first replicate of $condition\_1$, `cond1rep2` is the number of reads supporting replicate 2 in $condition\_1$, `cond2rep1` and `cond2rep2` are counts for replicates 1 and 2 of $condition\_2$.

2.1.3 Input table from *KisSplice* output

kissDE was developped to deal with *KisSplice* output, which is in fasta format. Below is the first four lines of an example of *KisSplice* output:

```
 headfasta <- system.file("extdata",
     "head_output_kissplice_alt_splicing_fasta.txt", package = "kissDE")
 writeLines(readLines(headfasta))
```

```
>bcc_68965|Cycle_4|Type_1|upper_path_length_112|AS1_1|SB1_1|S1_0|ASSB1_0|AS2_0|
SB2_0|S2_0|ASSB2_0|AS3_0|SB3_0|S3_0|ASSB3_0|AS4_1|SB4_0|S4_0|ASSB4_0|AS5_8|SB5_
2|S5_0|ASSB5_1|AS6_13|SB6_4|S6_0|ASSB6_3|AS7_4|SB7_1|S7_0|ASSB7_1|AS8_3|SB8_1|S
8_0|ASSB8_0|rank_0.76503
CACACCAGCCATAAAAAGCGAAAGAATAAAAACCGGCACAGCCCGTCTGGCATGTTTGATTATGACTTTGAGTATGTAT
ATTAGGTTAGGCTGGGAAGTTTTTTTTAAAAAC
>bcc_68965|Cycle_4|Type_1|lower_path_length_82|AB1_21|AB2_12|AB3_12|AB4_2|AB5_5
|AB6_1|AB7_2|AB8_1|rank_0.76503
CACACCAGCCATAAAAAGCGAAAGAATAAAAACCGGCACAGGTATGTATATTAGGTTAGGCTGGGAAGTTTTTTTTAAA
AAC
```

Events are reported in blocks of 4 lines, the first two lines correspond to one variant of the splicing event (or one allele of the SNV), the following two lines correspond to the other variant (or the other allele). As for all fasta file, there is a header line beginning with the `>` symbol and a line with the sequence. Each variant correspond to one entry in the fasta file.

Headers contain information used in *kissDE*. In the example, there are:

- elements shared by the headers of the two variants:
  - `bcc_68965|Cycle_4` is the event's ID.
  - `Type_1` means that the sequences correspond to a splicing event. `Type_0` corresponds to SNVs.
- elements that are specific to a variant:
  - `upper_path_length_112` and `lower_path_length_82` gives the length of the nucleotide sequences. Upper path and lower path are a denomination for the representation of each variant in *KisSplice*'s graph. For alternative splicing events, the upper path represents the inclusion isoform and the lower path the exclusion isoform.
  - `AS1_1|SB1_1|S1_0|ASSB1_0|AS2_0|SB2_0|S2_0|ASSB2_0|AS3_0| SB3_0|...` and `AB1_21|AB2_12|AB3_12|AB4_2|AB5_5|...` summarizes the counts found by *KisSplice* quantification step. Here *KisSplice* was run with the option `counts` set to 2. For the upper path, we have 4 counts for each sample: AS, SB, S and ASSB. For the lower path, we have 1 count per sample: AB. The different reads categories are shown on Figure 2. There are 8 sets of counts because we gave 8 files in input to *KisSplice* (denotated by the number before the "_" character). Each count (denotated by the number after the "_" character) corresponds to the reads coming from each file that could be mapped on the variant, in the order they have been passed to *KisSplice*.
- a rank information which is a deprecated measure.

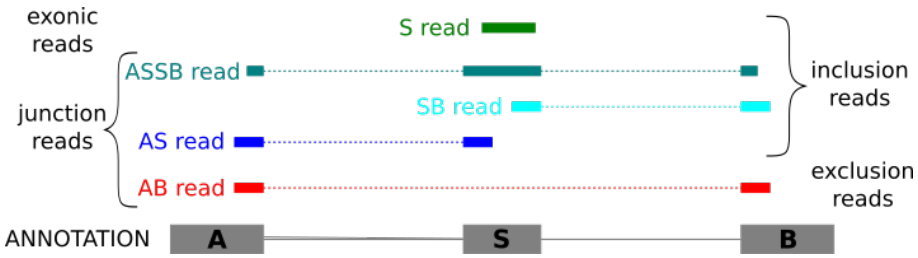

Figure 2: Different categories of reads
In this figure, we show an example of an alternative skipped exon. AS reads correspond to reads spanning the junction between the excluded sequence and its left flanking exon, SB to reads spanning the junction between the excluded sequence and its right flanking exon, ASSB to reads spanning the two inclusion junctions, S to reads entirely included in the alternative sequence and AB to reads spanning the junction between the two flanking exons. S reads correspond to exonic reads and all other categories of reads represented here correspond to junction reads.

kissDE can be used on any type of events output by *KisSplice* (0: SNV, 1: alternative splicing events, 3: indels,...). The user should refer to *KisSplice* manual (http://kissplice.prabi.fr/documentation/) for further questions about the *KisSplice* format and its output.

To be used in *kissDE*, *KisSplice* output must be converted into a table of counts. This can be done with the `kissplice2counts` function. In the example below, the *KisSplice* output file called '`output_kissplice_alt_splicing.fa`', included in the *kissDE* package, is loaded. The table of counts yielded by the `kissplice2counts` function is stored in `myCounts`.
Comment: `fpath2` contains the absolute path of the file on the user's hard disk.

```
fpath2 <- system.file("extdata", "output_kissplice_alt_splicing.fa",
    package = "kissDE")
myCounts <- kissplice2counts(fpath2, pairedEnd = TRUE)
```

The counts returned by `kissplice2counts` are extracted from the *KisSplice* header. By default, `kissplice2counts` expects single-end reads and one count for each variant.
The `counts` parameter of `kissplice2counts` must be the same as the `counts` parameter used to obtain data with *KisSplice*. The possible values are 0, 1 or 2. 0 is the default value for both `kissplice2counts` and *KisSplice*.
The user can also specify the `pairedEnd` parameter in `kissplice2counts`. If RNA-Seq libraries are paired-end, `pairedEnd` should be set to `TRUE`. In this case, the `kissplice2counts` function expects the counts of the paired-end reads to be next to each other. If it is not the case, an additional `order` parameter should be used to indicate the actual order of the counts. For instance, if the experimental design is composed of two conditions with two paired-end replicates and if the input in *KisSplice* followed this order:
cond1_sample1_readpair1, cond1_sample2_readpair1, cond2_sample1_readpair1, cond2_sample2_readpair1, cond1_sample1_readpair2, cond1_sample2_readpair2, cond2_sample1_readpair2 and cond2_sample2_readpair2.
The order vector should be equal to `c(1,2,3,4,1,2,3,4)`.
An example of a paired-end dataset run with `counts` equal to 0 is shown in section 4.2.

`kissplice2counts` returns a list of four elements, including `countsEvents` which contains the table of counts required in *kissDE*.

```
names(myCounts)
```

```
[1] "countsEvents"    "psiInfo"         "exonicReadsInfo" "k2rgFile"
```

```
head(myCounts$countsEvents)
```

```
        events.names events.length counts1 counts2 counts3 counts4
1  bcc_68965|Cycle_4           112       2       1      23       8
2  bcc_68965|Cycle_4            82      33      14       6       3
3  bcc_83285|Cycle_2           180     108      47      33      36
4  bcc_83285|Cycle_2            81       2       5     100     150
5 bcc_161433|Cycle_2           127      20      17      60      58
6 bcc_161433|Cycle_2            80      58      33       7       1
```

`myCounts$countsEvents` has the same structure as the `tableCounts` object in the section 2.1.2. It is a data frame with:

- **in rows:** One variation is represented by two lines, one for each variant. For instance for SNVs, one allele is described in the first line and the other in the second line. For alternative splicing events (as in this example), the inclusion and the exclusion isoform have one line each.
- **in columns:**
  - The first column (`events.names`) contains the name of the variation, using *KisSplice* notation.
  - The second column (`events.length`) contains the size of the variant in bp, extracted from the *KisSplice* header.
  - All others columns (`counts1, counts2, counts3, counts4`) contain counts for each replicate in each condition for the variant.

2.1.4 Input table from *KisSplice2refgenome* output

The `kissplice2counts` function can also deal with *KisSplice2refgenome* output data, in this case the `k2rg` parameter has to be set to `TRUE`. *KisSplice2refgenome* allows the annotation of the alternative splicing events. It assigns each event a gene and a type of alternative splicing event, among which: Exon Skipping (ES), Intron Retention (IR), Alternative Donor (AltD), Alternative Acceptor (AltA). Interested users should refer to *KisSplice2refgenome* manual for further questions about *KisSplice2refgenome* format and output (http://kissplice.prabi.fr/tools/kiss2refgenome/).

In the example below, 'output_k2rg_alt_splicing.txt', a *KisSplice2refgenome*'s output included in the *kissDE* package, is loaded. The `kissplice2counts` function uses the same `counts` and `pairedEnd` parameters as explained in the section 2.1.3. The table of counts yielded by the `kissplice2counts` function is stored in `myCounts_k2rg`. It has exactly the same structure as detailed in section 2.1.3.
Comment: `fpath3` contains the absolute path of the file on the user's hard disk.

```
fpath3 <- system.file("extdata", "output_k2rg_alt_splicing.txt",
    package = "kissDE")
myCounts_k2rg <- kissplice2counts(fpath3, pairedEnd = TRUE, k2rg = TRUE)
names(myCounts_k2rg)
```

```
[1] "countsEvents"    "psiInfo"         "exonicReadsInfo" "k2rgFile"
```

```
head(myCounts_k2rg$countsEvents)
```

```
        events.names events.length counts1 counts2 counts3 counts4
1 bcc_162707|Cycle_0           190      23      29      19      27
2 bcc_162707|Cycle_0            80     129     145     120     101
3 bcc_132936|Cycle_3           245     569     397     162     253
4 bcc_132936|Cycle_3            81      21       5       3       4
5 bcc_100903|Cycle_0           183      11      10       2       5
6 bcc_100903|Cycle_0            81     503     231     134     175
```

The *KisSplice2refgenome* output contains information about the type of splicing events. By default, all of the splicing events are analysed in *kissDE*, but it is also possible to focus on subtypes of events. This events selection will speed up *kissDE*'s running time and improve statistical power for choosen events. To do this, the `kissplice2counts` function contains two parameters: `keep` and `remove`. Both take a character vector indicating the types of events to keep or remove. The event names must be part of this list: `deletion`, `insertion`, `IR`, `ES`, `altA`, `altD`, `altAD`, `indel`, `-`.
Thus, if the user is only interested in intron retention events, the `keep` option should be set to `c("IR")`. If the user isn't interessed in deletions and insertions, the `remove` option should be equal to `c("insertion", "deletion")`.
The `keep` and `remove` parameters can be used at the same time only if `ES` is part of the `keep` vector. The `remove` vector will then act on the different types of exon skipping: multi-exon skipping (`MULTI`) or exon skipping associated with an alternative acceptor site (`altA`), an alternative donor site (`altD`), both alternative acceptor and donor site (`altAD`) or any of the alt combinaison with MULTI. Thus, in this specific case, the `remove` vector should contain names from this list: `MULTI`, `altA`, `altD`, `altAD`, `MULTI_altA`, `MULTI_altD`, `MULTI_altAD`.

If the user wants to analyse only cassette exon events (i.e., a single exon is skipped or included), the following command should be used:

```
myCounts_k2rg_ES <- kissplice2counts(fpath3, pairedEnd = TRUE,
    k2rg = TRUE, keep = c("ES"), remove = c("MULTI", "altA",
        "altD", "altAD", "MULTI_altA", "MULTI_altD", "MULTI_altAD"))
```

2.2 Quality Control

kissDE contains a function that allows the user to control the quality of the data and to check if no error occured at the data loading step. This data quality assessment is essential and should be done before the differential analysis.

The `qualityControl` function takes as input a count table (see sections 2.1.2, 2.1.3 and 2.1.4) and a condition vector (see section 2.1.1):

```
qualityControl(myCounts, myConditions)
```

It produces 2 graphs:

- a heatmap of the sample-to-sample distances using the 500 most variant events (see left panel of Figure 3)
- the factor map formed by the first two axes of a principal component analysis (PCA) using the 500 most variant events (see right panel of Figure 3)

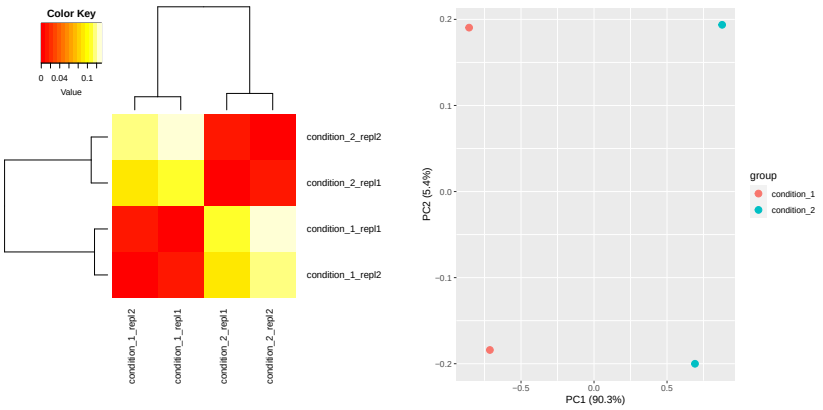

Figure 3: Quality control plots
Left: Heatmap of the sample-to-sample distances. Right: Principal Component Analysis.

These two graphs show the similarities and the differences between the analyzed samples. Replicates of the same condition are expected to cluster together. If this is not the case, the user should check if the order of the samples in the count table and in the condition vector is the same. If it is, this could mean that a sample is contaminated or has an abnormality that will influence the differential analysis. The user can then go back to the quality control of the raw data to solve the problem or decide to remove the sample from the analysis.

In the heatmap plot, the samples that cluster together are from the same condition. In the PCA plot, the first principal component (PC1) summarize 90.2% of the total variance of the dataset. This first axis clearly separates the 2 conditions.

The created graphs can be saved by setting the `storeFigs` parameter of the `qualityControl` function to `TRUE` (then graphs are stored in a 'kissDEFigures' folder, created in a temporary directory, which is removed at the end of the user *R* session) or to the path where the user wants to store his/her graphs. We recommend to use this parameter when the `qualityControl` function is used in an automatized workflow.

To customize the PCA plot, the data frame used for this plot can be extracted by setting the option `returnPCAdata` to `TRUE` as follows:

```
PCAdata <- qualityControl(myCounts, myConditions, returnPCAdata = TRUE)
```

2.3 Differential analysis

When data are loaded, the differential analysis can be run using the `diffExpressedVariants` function. This function has two mandatory parameters: a count table (`countsData` parameter, see sections 2.1.2, 2.1.3 and 2.1.4) and a condition vector (`conditions` parameter, see section 2.1.1).

In the example below, the differential analysis results are stored in the `myResults` object:

```
myResults <- diffExpressedVariants(countsData = myCounts,
    conditions = myConditions)
```

The `diffExpressedVariants` function has three parameters to change the filters or the flags applied on the data, one parameter to indicate if the replicates are technical or biological, and one parameter to indicate how many cores should be used :

- `pvalue`: By default, the p-value threshold to output the significant events is set to 1. So all variants are output in the final table. This parameter must be a numeric value between 0 and 1. Be aware that by setting `pvalue` to 0.05, only events that have been identified as significant between the conditions with a false discovery rate (FDR) $\leqslant 5\%$ will be present in the final table. A posteriori changing this threshold will require to re-run the differential analysis.
- `filterLowCountsVariants`: This parameter allows to change the threshold to filter low expressed events before testing (as explained in section 3.3). By default, it is set to 10.
- `flagLowCountsConditions`: This parameter allows to change the threshold to flag low expressed events (as explained in section 3.6). By default, it is set to 10.
- `technicalReplicates`: Boolean value indicating if the user is working with technical replicates only (we do not advise users to mix biological and technical replicates in their analyses). If this parameter is set to `TRUE`, the counts will be modeled with a Poisson distribution. If it is equal to `FALSE`, the counts will be modeled with a Negative Binomial distribution. For more information, see section 3.2. By default, this option is set to `FALSE`.
- `nbCore`: An integer value indicating how many cores should be used for the computation. This parameter should be strictly lower than the number of core of the computer (`nbCore` $<$ nbr computer cores -1). By default, this parameter is set to 1, meaning that the computation are not parallelized.

The `diffExpressedVariants` function returns a list of 6 objects:

```
names(myResults)
```

```
[1] "finalTable"          "correctedPVal"       "uncorrectedPVal"
[4] "resultFitNBglmModel" "f/psiTable"          "k2rgFile"
```

The `uncorrectedPVal` and `correctedPval` outputs are numeric vectors with p-values before and after correction for multiple testing. `resultFitNBglmModel` is a data frame containing the results of the fitting of the model to the data. `k2rgFile` is a string containing either the *KisSplice2refgenome* file path and name or NULL if no *KisSplice2refgenome* file was used as input. For explanations about the `finalTable` and `f/psiTable` outputs, see section 2.4.1 and section 2.4.2, respectively.

To visualize the distribution of the p-values before the application of the Benjamini-Hochberg [5] multiple testing correction procedure, the histogram of the p-values before correction can be plotted by using the following command:

```
hist(myResults$uncorrectedPVal, main = "Histogram of p-values",
    xlab = "p-values", breaks = 50)
```

Because the dataset used here is small ($\sim$ 100 lines), the histograms of the two complete datasets presented in the case studies (section 4) are represented. As expected, the histograms show a uniform distribution with a peak near 0 (Figure 4).

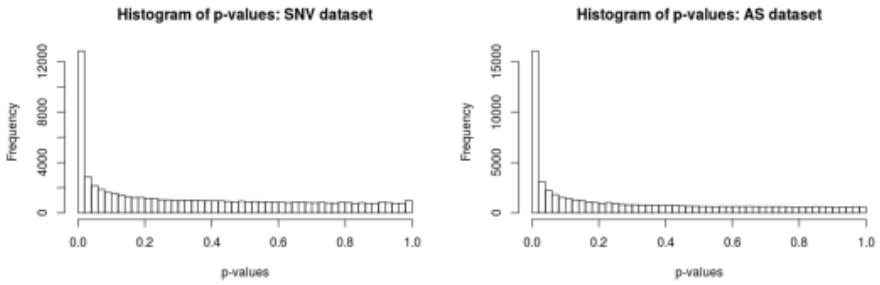

Figure 4: Distribution of p-values before correction for multiple testing
Left: for the complete dataset presented in section 4.2. Right: for the complete dataset presented in section 4.1.

2.4 Output results

2.4.1 Final table

The `finalTable` object is the main output of the `diffExpressedVariants` function. The first 3 rows of the `myResults$finalTable` output are as follows:

```
print(head(myResults$finalTable, n = 3), row.names = FALSE)
```

```
                    ID Length_diff Variant1_condition_1_repl1_Norm
      bcc_83285|Cycle_2          99                              86
      bcc_68965|Cycle_4          30                               2
bcc_135201|Cycle_433392         104                              80
Variant1_condition_1_repl2_Norm Variant1_condition_2_repl1_Norm
                             50                              36
```

```
                                1                                25
                               56                                31
Variant1_condition_2_repl2_Norm Variant2_condition_1_repl1_Norm
                               38                                 2
                                8                                26
                               35                                 2
Variant2_condition_1_repl2_Norm Variant2_condition_2_repl1_Norm
                                5                               108
                               15                                 6
                                1                                31
Variant2_condition_2_repl2_Norm Adjusted_pvalue Deltaf/DeltaPSI lowcounts
                              158        0.00e+00          -0.770     FALSE
                                3        7.93e-10           0.703     FALSE
                               58        0.00e+00          -0.696     FALSE
```

The columns of this table contain the following information:

- `ID` is the event identifier. Each event is represented by one row in the table.
- `Length_diff` contains the variable part length in a splicing event. It is the length difference between the upper and lower path. This column is not relevant for SNVs.
- `Variant1_condition_1_repl1_Norm` and following columns contain the counts for each replicate of each variant after normalization (raw counts are normalized as in the *DESeq2 Bioconductor R* package, see details in section 3.1). The first half of these columns concerns the first variant of each event, the second half the second variant.
- `Adjusted_pvalue` contains p-values adjusted by a Benjamini-Hochberg procedure.
- `Deltaf/DeltaPSI` summarizes the magnitude of the effect (see details in section 3.7).
- `lowcounts` contains booleans which flag low counts events as described in section 3.6. A `TRUE` value means that the event has low counts (counts below the chosen threshold).

In the `finalTable` output, events are sorted by p-values and then by magnitude of effect (based on their absolute values), so that the top candidates for further investigation/validation appear at the beginning of the output.

Warning: When the p-value computed by kissDE is lower than the smallest number greater than zero that can be stored (i.e., 2.2e-16), this p-value is set to 0.

To save results, a tab-delimited file can be written with `writeOutputKissDE` function where an `output` parameter (containing the name of the saved file) is required. Here, the `myResults` output is saved in a file called '`results_table.tab`':

```
writeOutputKissDE(myResults, output = "kissDE_results_table.tab")
```

Users can choose to export only events passing some thresholds on adjusted p-value and/or Deltaf/DeltaPSI using the options `adjPvalMax` and `dPSImin` of the `writeOutputKissDE` function. For example, if we want to save in a file called '`results_table_filtered.tab`' only events with the adjusted p-value $\leqslant 0.05$ and the Deltaf/DeltaPSI absolute value $\geqslant 0.10$, the following command can be used:

```
writeOutputKissDE(myResults, output = "kissDE_results_table_filtered.tab",
    adjPvalMax = 0.05, dPSImin = 0.1)
```

If the counts table was built from a *KisSplice2refgenome* output with the `kissplice2counts` function, running the `writeOutputKissDE` will write a file merging results of differential analysis with *KisSplice2refgenome* data. As previously explained (section 2.4.1), users can choose to save only events passing thresholds:

```
writeOutputKissDE(myResults_K2RG, output = "kissDE_K2RG_results_table.tab",
    adjPvalMax = 0.05, dPSImin = 0.1)
```

2.4.2 f/PSI table

The `f/psiTable` output of the `diffExpressedVariants` function contains the f values for SNV analysis or PSI values for alternative splicing analysis (see details and computation in section 3.7) for each event in each sample. The first three rows of the `f/psiTable` output of the `myResults` object (created in the section 2.3) look like this:

```
                  ID condition_1_repl1 condition_1_repl2 condition_2_repl1
1 bcc_100903|Cycle_0           0.00984            0.0195           0.00607
2 bcc_108176|Cycle_0           0.03805            0.0614           0.03844
3 bcc_120508|Cycle_0           0.94526            0.9477           0.96531
  condition_2_repl2
1            0.0119
2            0.0296
3            0.9414
```

This output can be useful to carry out downstream analysis or to produce specific plots (like heatmap on f/PSI events). To use this information with external tools, this table can be saved in a tab-delimited file (here called '`result_PSI.tab`'), setting the `writePSI` parameter to `TRUE` in the `writeOutputKissDE` function:

```
writeOutputKissDE(myResults, output = "result_PSI.tab", writePSI = TRUE)
```

3 *kissDE*'s theory

In this section, the different steps of the *kissDE* main function, `diffExpressedVariants`, are detailed. They are summarized in the Figure 5.

3.1 Normalization

In a first step, counts are normalized with the default normalization methods provided by the *DESeq2* [1] package. The size factors are estimated using the sum of counts of both variants for each event, which is a proxy of the gene expression. By using this normalization, we correct for library size, because the sequencing depth can vary between samples.

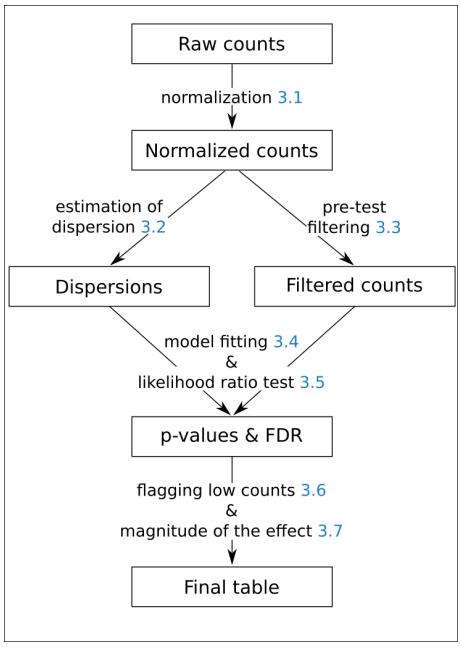

Figure 5: The different steps of the `diffExpressedVariants` function
Numbers in light blue point to the section of this vignette explaining the step.

3.2 Estimation of dispersion

A model to describe the counts distribution is first chosen. When working with technical replicates (`technicalReplicates` = `TRUE` in `diffExpressedVariants`), the Poisson model (model $\mathcal{M}(\phi = 0)$) is chosen in *kissDE*.
When working with biological replicates (`technicalReplicates` = `FALSE` in `diffExpressedVariants`), the Poisson distribution's variance parameter is in general not flexible enough to describe the data, because replicates add several sources of variance.
This overdispersion is often modeled using a Negative Binomial distribution. In *kissDE*, the overdispersion parameter, ϕ, is estimated using the *DSS* *R* package [6, 7, 8, 9] (model $\mathcal{M}(\phi = \phi^i_{\text{DSS}})$.
The *DSS* package (and, to our knowledge, every other package estimating the overdispersion of the Negative Binomial model) is suited for differential expression analysis (one count per sample). In differential splicing and SNV analysis, two counts (one for each splice variant or allele) are associated with each sample. In order to mimic gene expression, the overdispersion parameter ϕ is estimated on the sum of the splice variant or allele counts of each sample.

3.3 Pre-test filtering

If global counts for both variants are too low (option `filterLowCountsVariants`), the event is not tested. The rationale behind this filter is to speed up the analysis and gain statistical power.
Here we present an example to explain how `filterLowCountsVariants` option works. Let's assume that there are two conditions and two replicates per condition. `filterLowCountsVariants` keeps its default value, 10.

	Condition 1		Condition 2		Sum by variant
	replicate 1	replicate 2	replicate 1	replicate 2	
Variant 1	2	1	3	2	2+1+3+2=8 < 10
Variant 2	8	0	1	0	8+0+1+0=9 < 10

Table 1: Example of an event filtered out before the differential analysis, because less than 10 reads support each variant

In this example (Table 1), the two variants have global counts less than 10, this event will be used to compute the overdispersion, but will not be used to compute the models. It will neither appear in the result table.

3.4 Model fitting

Then we design two models to take into account interactions with variants (SNVs or alternative isoforms) and experimental conditions as main effects. We use the generalised linear model framework. The expected intensity λ_{ijk} can be written as follows:

$$\mathcal{M}_I : \quad \log \lambda_{ijk} = \mu + \alpha_i + \beta_j \tag{4}$$

$$\mathcal{M}_\infty : \quad \log \lambda_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} \tag{5}$$

where μ is the local mean expression of the transcript that contains the variant, α_i the effect of variant i on the expression, β_j the contribution of condition j to the total expression, and $(\alpha\beta)_{ij}$ the interaction term.

To avoid singular hessian matrices while fitting models, pseudo-counts (*i.e.*, systematic random allocation of ones) were considered for variants showing many zero counts.

3.5 Likelihood ratio test

To select between $\mathcal{M}_I$ and $\mathcal{M}_\infty$, we perform a Likelihood Ratio Test (LRT) with one degree of freedom. In the null hypothesis $H_0 : \{(\alpha\beta)_{ij} = 0\}$, there is no interaction between variant and condition. For events where H_0 is rejected, the interaction term is significant to explain the count's distribution, which leads to conclude to a differential usage of a variant across conditions. p-values are then adjusted with a 5% false discovery rate (FDR) following a Benjamini-Hochberg procedure [5] to account for multiple testing.

3.6 Flagging low counts

If in at least $n-1$ conditions (be n the number of conditions ≥ 2) an event has low counts (option `flagLowCountsConditions`), it is flagged (`TRUE` in the last column of the `finalTable` output).
In the example Table 2, we can see that the counts are quite contrasted, variant 1 seemed more expressed in condition 2 and variant 2 in condition 1. Moreover, this event has enough counts for each variant not to be filtered out when the `filterLowCountsVariants` parameter is set to 10:

	Condition 1		Condition 2		Sum by variant
	replicate 1	replicate 2	replicate 1	replicate 2	
Variant 1	1	0	60	70	1+0+60+70=131 $>$ 10
Variant 2	5	3	10	20	5+3+10+20=38 $>$ 10
Sum by condition	**9** $<$ 10		160 $>$ 10		

Table 2: Example of an event flagged as having low counts, because less than 10 reads support this event in the first condition

However, in $n-1$ (here 1) condition, the global count for one condition is less than 10 (9 for condition 1), so `flagLowCountsConditions` option will flag this event as `'Low_Counts'`. This event may be interesting because it has the potential to be found as differential. However, it will be hard to validate it experimentally, because the gene is poorly expressed in condition 1.

3.7 Magnitude of the effect

When a gene is found to be differentially spliced between two conditions, or an allele is found to be differentially present in two populations/conditions, one concern which remains is to quantify the magnitude of this effect. Indeed, especially in RNA-Seq, where some genes are very highly expressed (and hence have very high read counts), it is often the case that we detect significant (p-value $\leqslant 0.05$) but weak effects.

When dealing with genomic variants, we quantify the magnitude of the effect using the difference of allele frequencies (f) between the two conditions. When dealing with splicing variants, we quantify the magnitude of the effect using the difference of Percent Spliced In (PSI) between the two conditions. These two measures turn out to be equivalent and can be summarized using the following formula:

$$PSI = f = \frac{\#counts * \_variant_1}{\#counts * \_variant_1 + \#counts\_variant_2} \quad (6)$$

$$\Delta PSI = PSI_{cond1} - PSI_{cond2} \quad (7)$$

$$\Delta f = f_{cond1} - f_{cond2} \quad (8)$$

In this formula, $\#counts * \_variant_1$ correspond to the normalized number of reads of the $variant_1$, itself normalized for the variant length. Indeed, by construction, $variant_1$ always have a length greater than or equal to the $variant_2$. That's why we divide the normalized number of reads of the $variant_1$ by the ratio of the length of the $variant_1$ and the $variant_2$.

The ΔPSI/Δf is computed as follows:

- First, individual (per replicate) PSI/f are calculated. If counts for both upper and lower paths are too low (< 10) after normalization, the individual PSI/f are not computed.
- Then mean PSI/f are computed for each condition. If more than half of the individual PSI/f were not calculated at the previous step, the mean PSI/f is not computed either.
- Finally, we output ΔPSI/Δf. Unless one of the mean PSI/f of a condition could not be computed, ΔPSI/Δf is calculated subtracting one condition PSI/f from another. ΔPSI/Δf absolute value vary between 0 and 1, with values close to 0 indicating low

effects and values close to 1 strong effects. Note that the conditions are ordered alphabetically, and that *kissDE* substract the condition coming first in the alphabet to the other.

4 Case studies

To detect SNVs (SNPs, mutations, RNA editing) or alternative splicing (AS) in the expressed regions of the genome, *KisSplice* can be run on RNA-seq data. Counts can then be analysed using *kissDE*. We present two distinct case study with *kissDE*: analysis of AS events and analysis of SNVs.

4.1 Application of *kissDE* to alternative splicing

This first example corresponds to the case of differential analysis of alternative splicing (AS) events. The sample data presented here is a subset of the case study used in [3] (http://kissplice.prabi.fr/pipeline_ks_farline/).

4.1.1 Dataset

The data used in this example comes from the ENCODE project [10]. The samples are from a neuroblastoma cell line, SK-N-SH, with or without a retinoic acid treatment. Each condition is composed of two biological replicates. The data are paired-end.

In a preliminary step, *KisSplice* has been run to analyse these two conditions. Results from *KisSplice* (type 1 events) were then mapped to the reference genome with *STAR* [11] and analyzed with *KisSplice2refgenome*. *KisSplice2refgenome* enables to annotate the AS events discovered by *KisSplice*. It assigns to each event a gene and a type of alternative splicing (Exon Skipping (ES), Intron Retention (IR), Alternative Donor (AltD), Alternative Acceptor (AltA), ...).
For further information on these tools (*KisSplice* and *KisSplice2refgenome*), please refer to the manual that can be found on this web page: http://kissplice.prabi.fr/.

The output file of *KisSplice2refgenome* is a tab-delimited file that stores the annotated alternative splicing events found in the dataset. Below is an extract of this file (the first 3 rows and first 10 columns), where each row is one alternative splicing event of our data:

```
            Gene_Id Gene_name Chromosome_and_genomic_position Strand Event_type
ENSG00000066651.18    TRMT11           6:125999573-126008430      +         ES
ENSG00000124074.11     ENKD1            16:67663517-67666111      -         ES
ENSG00000112146.16     FBXO9             6:53065752-53071096      +         ES
Variable_part_length Frameshift_? CDS_?   Gene_biotype
                 110        False  True protein_coding
                 164        False  True protein_coding
                 102         True False protein_coding
             number_of_known_splice_sites/number_of_SNPs
3_ss_annotated_(over_4_ss):125999613,126006954,126008392
                           all_splice_sites_known_(4_ss)
                           all_splice_sites_known_(4_ss)
```

4.1.2 Load data

The `kissplice2counts` function allows to load directly the *KisSplice2refgenome* output file (here called '`output_k2rg_alt_splicing.txt`') into a format compatible with *kissDE*'s main functions.
Comment: `fileInAS` *contains the absolute path of the file on the user's hard disk.*
The `k2rg` parameter is set to `TRUE` to indicate that the file comes from *KisSplice2refgenome* and not directly from *KisSplice*. As these samples are paired-end, the `pairedEnd` parameter is set to `TRUE`. The `counts` parameter must be set to the same value (i.e., 2) used in *KisSplice* and *KisSplice2refgenome* to indicate which type of counts are given in the input. Here the exonic reads are not taken into account (`exonicReads = FALSE`). Only junction reads will be used (see Figure 2).

The table of counts is stored in a `myCounts_AS` object (for a detailed description of its structure, see section 2.1.4):

```
fileInAS <- system.file("extdata", "output_k2rg_alt_splicing.txt",
    package = "kissDE")
myCounts_AS <- kissplice2counts(fileInAS, pairedEnd = TRUE, k2rg = TRUE,
    counts = 2, exonicReads = FALSE)
head(myCounts_AS$countsEvents)
```

```
       events.names events.length counts1 counts2 counts3 counts4
1 bcc_162707|Cycle_0           190      17      24      13      20
2 bcc_162707|Cycle_0            80     129     145     120     101
3 bcc_132936|Cycle_3           245     324     177      64      88
4 bcc_132936|Cycle_3            81      21       5       3       4
5 bcc_100903|Cycle_0           183      10       9       1       3
6 bcc_100903|Cycle_0            81     503     231     134     175
```

To perform the differential analysis, a vector that describes the experimental plan is needed. In this case study, there are two replicates of the SK-N-SH cell line without treatment (SKNSH) followed by two replicates of the same cell line treated with retinoic acid (SKSNH-RA). So the `myConditions_AS` vector is defined as follows:

```
myConditions_AS <- c(rep("SKNSH", 2), rep("SKNSH-RA", 2))
```

4.1.3 Quality control

Before running the differential analysis, we check that the data was loaded correctly, using the `qualityControl` function.

```
qualityControl(myCounts_AS, myConditions_AS)
```

On both plots returned by the `qualityControl` function (Figure 6), the replicates of the same condition seem to be more similar between themselves than to the samples of the other condition. On the heatmap (left of Figure 6), the samples of the same condition cluster together. On the PCA plot (right of Figure 6), the first principal component (which summarises 88% of the total variance) clearly discriminates the two conditions.

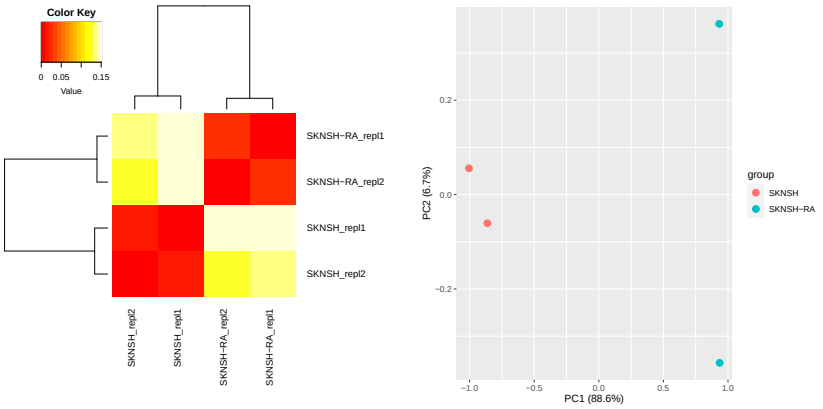

Figure 6: Quality control plots on alternative data
Left: Heatmap of the sample-to-sample distances for the alternative splicing dataset. Right: Principal Component Analysis for the alternative splicing dataset.

4.1.4 Differential analysis

The main function of *kissDE*, `diffExpressedVariants`, can now be run to compute the differential analysis. Outputs are stored in a `myResult_AS` object (for a detailed description of its structure, see section 2.4.1) and the result for the first three events is given below:

```
myResult_AS <- diffExpressedVariants(myCounts_AS, myConditions_AS)
head(myResult_AS$finalTable, n = 3)
```

```
                                        ID Length_diff
bcc_83285|Cycle_2             bcc_83285|Cycle_2          99
bcc_52250|Cycle_0             bcc_52250|Cycle_0         160
bcc_135201|Cycle_433392 bcc_135201|Cycle_433392         104
                        Variant1_SKNSH_repl1_Norm Variant1_SKNSH_repl2_Norm
bcc_83285|Cycle_2                              84                         44
bcc_52250|Cycle_0                              10                         24
bcc_135201|Cycle_433392                        40                         29
                        Variant1_SKNSH-RA_repl1_Norm
bcc_83285|Cycle_2                                 17
bcc_52250|Cycle_0                                 15
bcc_135201|Cycle_433392                           19
                        Variant1_SKNSH-RA_repl2_Norm Variant2_SKNSH_repl1_Norm
bcc_83285|Cycle_2                                 28                          2
bcc_52250|Cycle_0                                 14                          2
bcc_135201|Cycle_433392                           27                          2
                        Variant2_SKNSH_repl2_Norm Variant2_SKNSH-RA_repl1_Norm
bcc_83285|Cycle_2                               5                           110
bcc_52250|Cycle_0                               0                            19
bcc_135201|Cycle_433392                         1                            32
                        Variant2_SKNSH-RA_repl2_Norm Adjusted_pvalue
bcc_83285|Cycle_2                                162        0.00e+00
bcc_52250|Cycle_0                                 24        1.63e-06
bcc_135201|Cycle_433392                           59        1.88e-13
                        Deltaf/DeltaPSI lowcounts
bcc_83285|Cycle_2                -0.809     FALSE
```

```
bcc_52250|Cycle_0                -0.746     FALSE
bcc_135201|Cycle_433392          -0.715     FALSE
```

The first event in the `myResult_AS` output has a very low p-value (`Adjusted_pvalue` column, less than `2.2e-16`) and a very contrasted ΔPSI (`Deltaf/DeltaPSI` column, equal to `-0.804`) close to the maximum value (1 in absolute). This gene is differentially spliced. When the SK-N-SH cell line is treated with retinoic acid, the inclusion variant becomes the major isoform.

4.1.5 Export results

In order to facilitate the downstream analysis of the results, two tables are exported: the result table (`myResults_AS$finalTable` object, see section 2.4.1) is saved in a 'results_table.tab' file and the PSI table (`myResults_AS$\`f/psiTable\``, see section 2.4.2) is saved in a 'psi_table.tab' file. Here are the commands to carry out this task:

```
writeOutputKissDE(myResults_AS, output = "results_table.tab")
writeOutputKissDE(myResults_AS, output = "psi_table.tab", writePSI = TRUE)
```

4.2 Application of *kissDE* to SNPs/SNVs

This second example present an analysis of SNPs/SNVs done with *kissDE* on RNA-Seq data from a subset of the case study presented in [2] (http://kissplice.prabi.fr/TWAS/).
The original purpose of this study was to demonstrate that the method can deal with pooled data (i.e. individuals are pooled prior to sequencing). Pooling can be used to decrease the costs. It is also sometimes the only option, when too few RNA is available per individual. The method can in principle be used on unpooled data, polyploid genomes, and for the detection of somatic mutations, but has for now only been evaluated for the detection of SNPs/SNVs in pooled RNAseq data.
In the remaining, we use the term SNV, which designates a variation of a single nucleotide, without any restriction on the frequency of the two alleles. The term SNP is indeed classically used for variants present in at least 1% of a population.

4.2.1 Dataset

The dataset comes from the human GEUVADIS project. Two populations were selected: Toscans (TSC) and Central Europeans (CEU). For each population, we selected 10 individuals, which are pooled in two groups of 5. Each group corresponds to a replicate for *kissDE*. The conditions being compared are the populations.

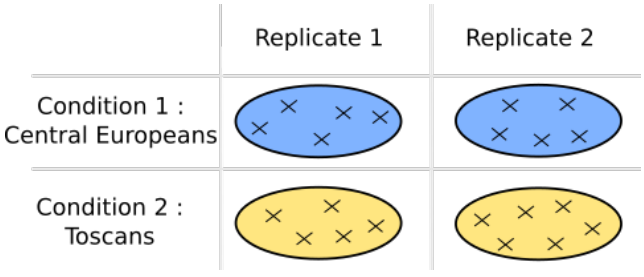

Figure 7: Experimental design of the SNP dataset
Each cross corresponds to an individual.

The data are paired-end. So each sample consists of 2 files. In total, 8 files have been used: 4 files for the two TSC samples and 4 files for the two CEU samples. Paired-end files from a same sample have been given as following each other to *KisSplice*.

KisSplice outputs a fasta file that stores SNVs found in the dataset. Its structure is described in section 2.1.3. The first SNV is presented below:

```
>bcc_44787|Cycle_421687|Type_0b|upper_path_length_131|C1_455|C2_455|C3_839|
C4_848|C5_5|C6_0|C7_39|C8_31|Q1_58|Q2_55|Q3_51|Q4_53|Q5_70|Q6_0|Q7_66|Q8_65|
rank_0.97008
CCAGAGAATCGGTCAGGGACCCCTGAGGGCCGCTGATTATTCCTATAGATGAGGAGTTTGGGGGCCGTTCCTGGGA
GCTGCTGGTACCAGTTTACAGTATTACTTCCGATGTTGGAGCTGCTTCCAGAACA
>bcc_44787|Cycle_421687|Type_0b|lower_path_length_131|C1_12|C2_14|C3_11|
C4_11|C5_18|C6_10|C7_4481|C8_4088|Q1_0|Q2_0|Q3_0|Q4_0|Q5_0|Q6_0|Q7_35|Q8_35|
rank_0.97008
CCAGAGAATCGGTCAGGGACCCCTGAGGGCCGCTGATTATTACTAGAGAAGAGGAGTTTGGGGGCCGTTCCTGGGA
GCTGCTGGTACCAATTTACAGTATTACTTCCGATGTTGGAGCTGCTTCCAGAACA
```

Events are reported in 4 lines, the two first represent one allele of the SNV, the two last the other allele. Thus the sequences only differ from each other at one position which corresponds to the SNV, here A/C in the center of the sequence (at position 42).

Because *KisSplice* was run with the default value of the `counts` parameter (i.e., 0), the counts have the following format `C1_x|C2_y|...|Cn_z`. In this example, there are 8 counts because we input 8 files. Each count corresponds to the reads coming from each file that could be mapped on the variant, in the order they have been passed to *KisSplice*. This information is particularly important in *kissDE* since it represents the counts used for the test.

4.2.2 Load data

The first step is to convert this fasta file (here called 'output_kissplice_SNV.fa') into a format that will be used in *kissDE* main functions, thanks to the `kissplice2counts` function.
Comment: `fileInSNV` contains the absolute path of the file on the user's hard disk.
Due to paired-end RNA-Seq data, the `pairedEnd` parameter was set to `TRUE`.

This conversion in a table of counts is stored in the `myCounts_SNV` object (for a detailed description of its structure, see section 2.1.3) and can be done as follows:

```
fileInSNV <- system.file("extdata", "output_kissplice_SNV.fa",
    package = "kissDE")
myCounts_SNV <- kissplice2counts(fileInSNV, counts = 0, pairedEnd = TRUE)
head(myCounts_SNV$countsEvents)
```

```
            events.names events.length counts1 counts2 counts3 counts4
1 bcc_44787|Cycle_421687           131     910    1687       5      70
2 bcc_44787|Cycle_421687           131      26      22      28    8569
3 bcc_44787|Cycle_421701           139     389    3349       2     149
4 bcc_44787|Cycle_421701           139      88      31      29    8821
5     bcc_100871|Cycle_3           107       0      10       0       0
6     bcc_100871|Cycle_3           107       3       1      13      10
```

To perform the differential analysis, a vector with the conditions has to be provided.
In the example, there are two replicates of TSC and two replicates of CEU, thus the condition vector `myConditions_SNV` is:

```
myConditions_SNV <- c(rep("TSC", 2), rep("CEU", 2))
```

4.2.3 Quality control

Before running the differential analysis, we recommand to check if the data was correctly loaded, by running the `qualityControl` function.

```
qualityControl(myCounts_SNV, myConditions_SNV)
```

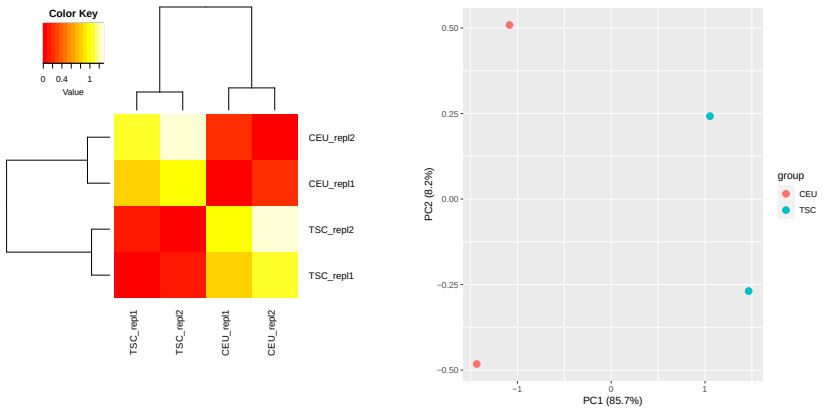

Figure 8: Quality control plots on SNV data
Left: Heatmap of the sample-to-sample distances on SNV data. Right: Principal Component Analysis on SNV data.

On both plots outputed (Figure 8), the replicates of the same condition seem to be more similar between themselves than to the samples of the other condition. On the heatmap (left of Figure 8), the samples of the same condition cluster together. On the PCA plot (right of Figure 8), the first principal component (which summarises 88% of the total variance) clearly discriminates the two conditions.

4.2.4 Differential analysis

The main function of *kissDE*, `diffExpressedVariants`, can now be run to compute the statistical test.
Outputs are stored in a `myResult_SNV` object (for a detailed description of its structure, see section 2.4.1) and the result for the first three events is printed:

```
myResult_SNV <- diffExpressedVariants(myCounts_SNV, myConditions_SNV)
head(myResult_SNV$finalTable, n = 3)
```

```
                                        ID Length_diff
bcc_44787|Cycle_320265 bcc_44787|Cycle_320265           0
bcc_100871|Cycle_3         bcc_100871|Cycle_3           0
bcc_44787|Cycle_421687 bcc_44787|Cycle_421687           0
                       Variant1_CEU_repl1_Norm Variant1_CEU_repl2_Norm
bcc_44787|Cycle_320265                    2014                    1172
bcc_100871|Cycle_3                           0                       0
bcc_44787|Cycle_421687                       5                      72
                       Variant1_TSC_repl1_Norm Variant1_TSC_repl2_Norm
```

```
                        0                      2
bcc_100871|Cycle_3                          0                     10
bcc_44787|Cycle_421687                    959                   1672
                        Variant2_CEU_repl1_Norm Variant2_CEU_repl2_Norm
bcc_44787|Cycle_320265                     23                    181
bcc_100871|Cycle_3                         12                     10
bcc_44787|Cycle_421687                     25                   8836
                        Variant2_TSC_repl1_Norm Variant2_TSC_repl2_Norm
bcc_44787|Cycle_320265                    179                    853
bcc_100871|Cycle_3                          3                      1
bcc_44787|Cycle_421687                     27                     22
                        Adjusted_pvalue Deltaf/DeltaPSI lowcounts
bcc_44787|Cycle_320265        0.00e+00          -0.926     FALSE
bcc_100871|Cycle_3            1.46e-04           0.909     FALSE
bcc_44787|Cycle_421687        1.85e-05           0.892     FALSE
```

The first event in the `myResult_SNV` output has a low p-value (`Adjusted_pvalue` column, equal to `8.63e-13`) and a very high absolute value of Δf (`Deltaf/DeltaPSI` column, equal to `-0.926`) close to the maximum value (1 in absolute). This SNP would typically be population specific. One allele is enriched in the Toscan population, the other in the European population.

4.2.5 Export results

We consider as significant the events that have an adjusted p-value lower than 5%, so we set `adjPvalMax = 0.05`. Results passing this threshold are saved in a '`final_table_significants.tab`' file, with the `writeOutputKissDE` function, as follows:

```
writeOutputKissDE(myResults_SNV, output = "final_table_significants.tab",
    adjPvalMax = 0.05)
```

4.3 Time / Requirements

The statistical analysis function (`diffExpressedVariants`) is the most time-consuming steps. Here is an example of the running time of this function on the two complete datasets presented in the case studies(section 4). The time presented were evaluated on a desktop computer with the following caracteristics: Intel Core i7, CPU 2,60 GHz, 16G RAM.

Dataset	Options	Number of events	Running time of `diffExpressedVariants`
AS data	`counts=2`, `pairedEnd=TRUE` `k2rg=TRUE`	59132	17m
SNV data	`counts=0`, `pairedEnd=TRUE`	64824	18m

Table 3: Profiling
Running time of the principal function of *kissDE* (`diffExpressedVariants`) for two datasets (AS dataset from the ENCODE project [10] described in section 4.1 and SNV dataset from the GEUVADIS project [12] described in section 4.2).

To reduce even more the running time of `diffExpressedVariants`, the parameter `nbCore` can be used to parallelize the most time-consuming step of this function (for more detailed explanation on this parameter see section 2.3).

5 Session info

```
 sessionInfo()

R version 4.0.0 (2020-04-24)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server 2012 R2 x64 (build 9600)

Matrix products: default

locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] kissDE_1.8.0

loaded via a namespace (and not attached):
 [1] Biobase_2.48.0              foreach_1.5.0
 [3] bit64_0.9-7                 splines_4.0.0
 [5] R.utils_2.9.2               DelayedMatrixStats_1.10.0
 [7] gtools_3.8.2                assertthat_0.2.1
 [9] BiocManager_1.30.10         stats4_4.0.0
[11] blob_1.2.1                  BSgenome_1.56.0
[13] GenomeInfoDbData_1.2.3      Rsamtools_2.4.0
[15] yaml_2.2.1                  pillar_1.4.3
[17] RSQLite_2.2.0               lattice_0.20-41
[19] glue_1.4.0                  limma_3.44.0
[21] digest_0.6.25               GenomicRanges_1.40.0
[23] RColorBrewer_1.1-2          XVector_0.28.0
[25] colorspace_1.4-1            R.oo_1.23.0
[27] htmltools_0.4.0             Matrix_1.2-18
[29] DESeq2_1.28.0               XML_3.99-0.3
[31] pkgconfig_2.0.3             DSS_2.36.0
[33] genefilter_1.70.0           zlibbioc_1.34.0
[35] purrr_0.3.4                 xtable_1.8-4
[37] scales_1.1.0                gdata_2.18.0
[39] HDF5Array_1.16.0            BiocParallel_1.22.0
[41] tibble_3.0.1                annotate_1.66.0
[43] farver_2.0.3                IRanges_2.22.0
```

```
[45] ggplot2_3.3.0               ellipsis_0.3.0
[47] SummarizedExperiment_1.18.0 BiocGenerics_0.34.0
[49] survival_3.1-12             magrittr_1.5
[51] crayon_1.3.4                memoise_1.1.0
[53] evaluate_0.14               R.methodsS3_1.8.0
[55] doParallel_1.0.15           gplots_3.0.3
[57] bsseq_1.24.0                tools_4.0.0
[59] data.table_1.12.8           BiocStyle_2.16.0
[61] lifecycle_0.2.0             matrixStats_0.56.0
[63] Rhdf5lib_1.10.0             S4Vectors_0.26.0
[65] munsell_0.5.0               locfit_1.5-9.4
[67] DelayedArray_0.14.0         AnnotationDbi_1.50.0
[69] Biostrings_2.56.0           aod_1.3.1
[71] compiler_4.0.0              GenomeInfoDb_1.24.0
[73] caTools_1.18.0              rlang_0.4.5
[75] rhdf5_2.32.0                grid_4.0.0
[77] RCurl_1.98-1.2              iterators_1.0.12
[79] labeling_0.3                bitops_1.0-6
[81] rmarkdown_2.1               codetools_0.2-16
[83] gtable_0.3.0                DBI_1.1.0
[85] R6_2.4.1                    GenomicAlignments_1.24.0
[87] rtracklayer_1.48.0          knitr_1.28
[89] dplyr_0.8.5                 bit_1.1-15.2
[91] KernSmooth_2.23-17          permute_0.9-5
[93] parallel_4.0.0              Rcpp_1.0.4.6
[95] vctrs_0.2.4                 geneplotter_1.66.0
[97] tidyselect_1.0.0            xfun_0.13
```

References

[1] Michael I. Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12):550, 2014. doi:10.1186/s13059-014-0550-8.

[2] Hélène Lopez-Maestre, Lilia Brinza, Camille Marchet, Janice Kielbassa, Sylvère Bastien, Mathilde Boutigny, David Monnin, Adil El Filali, Claudia Marcia Carareto, Cristina Vieira, Franck Picard, Natacha Kremer, Fabrice Vavre, Marie-France Sagot, and Vincent Lacroix. SNP calling from RNA-seq data without a reference genome: identification, quantification, differential analysis and impact on the protein sequence. *Nucleic Acids Research*, 44(19):e148, 2016. doi:10.1093/nar/gkw655.

[3] Clara Benoit-Pilven, Camille Marchet, Emilie Chautard, Leandro Lima, Marie-Pierre Lambert, Gustavo Sacomoto, Amandine Rey, Cyril Bourgeois, Didier Auboeuf, and Vincent Lacroix. Annotation and differential analysis of alternative splicing using de novo assembly of rnaseq data. *bioRxiv*, 2016. URL: https://www.biorxiv.org/content/early/2016/09/12/074807, arXiv:https://www.biorxiv.org/content/early/2016/09/12/074807.full.pdf, doi:10.1101/074807.

[4] Gustavo A. T. 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*, 13(6):S5, 2012. doi:10.1186/1471-2105-13-S6-S5.

[5] Yoav Benjamini and Yosef Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995. URL: http://www.jstor.org/stable/2346101.

[6] Hao Wu, Chi Wang, and Zhijin Wu. A new shrinkage estimator for dispersion improves differential expression detection in RNA-seq data. *Biostatistics*, 14(2):232–43, 2013. doi:10.1093/biostatistics/kxs033.

[7] Hao Feng, Karen N. Conneely, and Hao Wu. A Bayesian hierarchical model to detect differentially methylated loci from single nucleotide resolution sequencing data. *Nucleic Acids Research*, 42(8):e69, 2014. doi:10.1093/nar/gku154.

[8] Hao Wu, Tianlei Xu, Hao Feng, Li Chen, Ben Li, Bing Yao, Zhaohui Qin, Peng Jin, and Karen N. Conneely. Detection of differentially methylated regions from whole-genome bisulfite sequencing data without replicates. *Nucleic Acids Research*, 43(21):e141, 2015. doi:10.1093/nar/gkv715.

[9] Yongseok Park and Hao Wu. Differential methylation analysis for BS-seq data under general experimental design. *Bioinformatics*, 32(10):1446, 2016. doi:10.1093/bioinformatics/btw026.

[10] Sarah Djebali, Carrie A. Davis, Angelika Merkel, Alex Dobin, Timo Lassmann, Ali Mortazavi, Andrea Tanzer, Julien Lagarde, Wei Lin, Felix Schlesinger, Chenghai Xue, Georgi K. Marinov, Jainab Khatun, Brian A. Williams, Chris Zaleski, Joel Rozowsky, Maik Roder, Felix Kokocinski, Rehab F. Abdelhamid, Tyler Alioto, Igor Antoshechkin, Michael T. Baer, Nadav S. Bar, Philippe Batut, Kimberly Bell, Ian Bell, Sudipto Chakrabortty, Xian Chen, Jacqueline Chrast, Joao Curado, Thomas Derrien, Jorg Drenkow, Erica Dumais, Jacqueline Dumais, Radha Duttagupta, Emilie Falconnet, Meagan Fastuca, Kata Fejes-Toth, Pedro Ferreira, Sylvain Foissac, Melissa J. Fullwood, Hui Gao, David Gonzalez, Assaf Gordon, Harsha Gunawardena, Cedric Howald, Sonali Jha, Rory Johnson, Philipp Kapranov, Brandon King, Colin Kingswood, Oscar J. Luo, Eddie Park, Kimberly Persaud, Jonathan B. Preall, Paolo Ribeca, Brian Risk, Daniel Robyr, Michael Sammeth, Lorian Schaffer, Lei-Hoon See, Atif Shahab, Jorgen Skancke, Ana Maria Suzuki, Hazuki Takahashi, Hagen Tilgner, Diane Trout, Nathalie Walters, Huaien Wang, John Wrobel, Yanbao Yu, Xiaoan Ruan, Yoshihide Hayashizaki, Jennifer Harrow, Mark Gerstein, Tim Hubbard, Alexandre Reymond, Stylianos E. Antonarakis, Gregory Hannon, Morgan C. Giddings, Yijun Ruan, Barbara Wold, Piero Carninci, Roderic Guigo, and Thomas R. Gingeras. Landscape of transcription in human cells. *Nature*, 489(7414):101–108, 2012. doi:10.1038/nature11233.

[11] Alexander Dobin, Carrie A. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R. Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013. doi:10.1093/bioinformatics/bts635.

[12] Tuuli Lappalainen, Michael Sammeth, Marc R. Friedländer, Peter A. C. 't Hoen, Jean Monlong, Manuel A. Rivas, Mar Gonzàlez-Porta, Natalja Kurbatova, Thasso Griebel, Pedro G. Ferreira, Matthias Barann, Thomas Wieland, Liliana Greger, Maarten van Iterson, Jonas Almlöf, Paolo Ribeca, Irina Pulyakhina, Daniela Esser, Thomas Giger,

Andrew Tikhonov, Marc Sultan, Gabrielle Bertier, Daniel G. MacArthur, Monkol Lek, Esther Lizano, Henk P. J. Buermans, Ismael Padioleau, Thomas Schwarzmayr, Olof Karlberg, Halit Ongen, Helena Kilpinen, Sergi Beltran, Marta Gut, Katja Kahlem, Vyacheslav Amstislavskiy, Oliver Stegle, Matti Pirinen, Stephen B. Montgomery, Peter Donnelly, Mark I. McCarthy, Paul Flicek, Tim M. Strom, The Geuvadis Consortium, Hans Lehrach, Stefan Schreiber, Ralf Sudbrak, Angel Carracedo, Stylianos E. Antonarakis, Robert Häsler, Ann-Christine Syvänen, Gert-Jan van Ommen, Alvis Brazma, Thomas Meitinger, Philip Rosenstiel, Roderic Guigó, Ivo G. Gut, Xavier Estivill, and Emmanouil T. Dermitzakis. Transcriptome and genome sequencing uncovers functional variation in humans. *Nature*, 501(7468):506–11, 2013. doi:10.1038/nature12531.