kissDE Reference Manual

Clara Benoit-Pilven, Camille Marchet, Janice Kielbassa, Audric Cologne, Aurélie Siberchicot, Vincer

May 12, 2017

Abstract

kissDE is a package dedicated to the analysis of count data obtained from quantification of pairs of variants in RNAseq data.
It can be used to study splice variants, where the two variant of the pair differ by the inclusion/exclusion of an exonic 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 similar to `DESeq`. 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. (Lopez-Maestre et al., 2016), the workflow for splicing is fully described in Benoit-Pilven et al.

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.
The input count data to **kissDE** may come from **KisSplice**(Sacomoto et al., 2012), **KisSplice2refgenome** or any other software which produces count data properly formatted. If you use the output files from **KisSplice** or **KisSplice2refgenome**, you can directly import your counts using `kissplice2counts` function.
The user must keep in mind that **kissDE** was designed to work with at least two replicates for each condition, which means that the minimal input contains the variants quantified for 4 different samples, each couple representing a condition and its 2 replicates. There can be more replicates and more conditions. It is not mandatory to have an equal number of replicates in each condition.

1.2 Install and load kissDE

In a **R** session, you must first install **BioConductor**.

```
> source("http://bioconductor.org/biocLite.R")
```

Then you can install the **kissDE** package from **BioConductor**.

```
> biocLite("kissDE")
```

Finally, load it.

```
> library(kissDE)
```

1.3 Quick start

Here we present the basic **R** commands for an analysis with **kissDE**. These commands require an example output file of **KisSplice** called `output_kissplice.fa`. To deal with other types of input file, please refer to section 2.1. The used funtions `kissplice2counts`, `qualityControl`, `diffExpressedVariants` and `writeOutputKissDE` of **kissDE** are detailed in section 2. Here we assume that there are two conditions (`condition_1` and `condition_2`) with two biological replicates and 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, adjPvalMax = 0.05, dPSImin = 0.1,
     output = "kissDE_output.tab")
```

2 kissDE's workflow

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

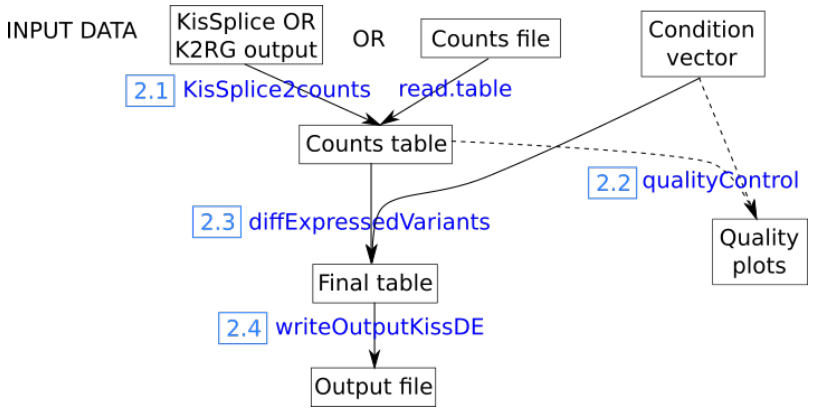

Figure 1: **Schema of kissDE's workflow. Numbers in light blue point to the section of this vignette explaining the step.**

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. Three options are possible, either to provide a table of counts or to work with **KisSplice**'s output or with **KisSplice2refgenome**'s output.

2.1.1 Condition vector

The first item to prepare is the condition vector that describes the order of the columns in the count table. Considering an example where you have 2 conditions and 2 replicates per condition, the condition vector will be:

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

because the counts have been input in an order that represents first the two replicates of *condition*_1, then the two replicates of *condition*_2.
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, you will get an error message.

2.1.2 You work with your 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` (`fpath1` contains the absolute path of the file on your hard disk) is loaded and stored in a `tableCounts` object.

```
> 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.
  - A head row must contain the columns name in the flat file.
- **In columns:**
  - The first column (`eventsName`) contains the name of the variation.

- The second column (`eventsLength`) contains the variant size in nucleotide (bp). If this size is unknown, the value can be set to 0. Note that the size is used only in the context of alternative splicing. Then it indicates whether the size difference due to the switch from one isoform to the other is a multiple of 3 or not and thus if it can induce a frameshift in the open reading frame (ORF).
- All other columns (`cond1rep1, cond1rep2, cond2rep1, cond2rep2`) contain read counts of a variant in each 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 sequenced for 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 can also deal with **KisSplice** output, which is in fasta format. Below is an example of the four first lines of a **kissDE** output:

```
>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 two first are about one variant of the splicing event (or one allele of the SNV), the following two lines about the other variant (or the other allele). For each variant, there is a header beginning with the `>` symbol and a sequence.

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 correspondto a splicing event. `Type_0` would correspond 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|...` and `AB1_21|AB2_12|AB3_12|AB4_2|AB5_5|...` summarizes the counts found by **KisSplice** during the step where reads are used to perform quantification of each variant detected. Here **KisSplice** was run with the option `counts` set to 2. For the upper path, we have 4 counts for each sample (the different reads categories are shown on Figure 2), and for the lower path, we have 1 count per sample. There are 8 sets of counts because we gave 8 files in input to **KisSplice**. 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**.

- a rank information which is a deprecated measure.

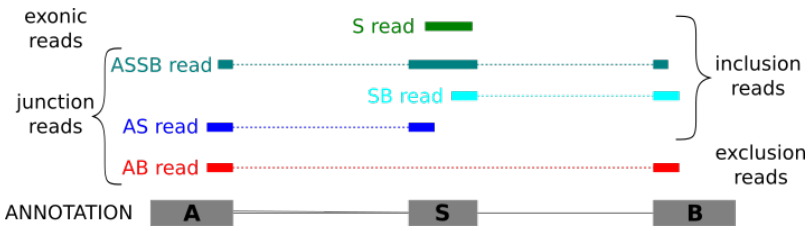

Figure 2: **Different categories of reads:** In this figure, we show an example of an alternative skipped exon. AS corresponds to reads spanning the junction between the excluded sequence and its left flanking exon, SB to reads spanning the junction between the excluded sequence and its right flanking 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 output from **KisSplice** (0: SNV, 1: alternative splicing events, 3: indels,...). The user should refer to **KisSplice** manual (`http://kissplice.prabi.fr/documentation/user_guide.pdf`) for further questions about **KisSplice** format and output.

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

```
> fpath2 <- system.file("extdata", "output_kissplice_alt_splicing.fa",
     package = "kissDE")
> myCounts <- kissplice2counts(fpath2, counts = 2, 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. If this is not the case, some parameters need to be adapted.
First, 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**.
Then, the user should also specify the `pairedEnd` parameter in `kissplice2counts`. If RNA-Seq libraries are single-end, `pairedEnd` should be set to `FALSE` (the default value). 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.
An example of a paired-end dataset run with `counts` equal to 0 is shown in section 4.1.

`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 of 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. It indicates whether the size difference due to the switch from one isoform to the other is a multiple of 3 or not (thus induces a frameshift in the ORF).
- 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, thanks to the `k2rg` parameter. **KisSplice2refgenome** allows the annotation of the alternative splicing events. It assigns each event to the gene it maps to and to a type of alternative splicing event: Skipped Exon (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 (ftp://pbil.univ-lyon1.fr/pub/logiciel/kissplice/tools/kissplice2refgenome_userguide.pdf).

In the example below, a **KisSplice2refgenome**'s output recorded in a file called `output_k2rg_alt_splicing.txt` (`fpath3` contains the absolute path of the file on your hard disk) is loaded. The `kissplice2counts` function here uses the same `counts` and `pairedEnd` parameters as explained in the section 2.1.3. The `k2rg` parameter is a boolean value, set to `TRUE` to deal with **KisSplice2refgenome** output data. The table of counts yielded by the `kissplice2counts` function in the example below is stored in `myCounts_k2rg`. Its has exactly the same structure as detailed in section 2.1.3.

```
> fpath3 <- system.file("extdata", "output_k2rg_alt_splicing.txt",
     package = "kissDE")
> myCounts_k2rg <- kissplice2counts(fpath3, counts = 2, 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_68965|Cycle_4            30       2       1      23       8
2  bcc_68965|Cycle_4            82      33      14       6       3
3  bcc_83285|Cycle_2            99     108      47      33      36
```

```
4  bcc_83285|Cycle_2            81       2       5     100     150
5 bcc_161433|Cycle_2            47      20      17      60      58
6 bcc_161433|Cycle_2            80      58      33       7       1
```

The **KisSplice2refgenome** output contains information about the type of splicing events. It is therefore possible to choose the subtype of events you want to analyse. The `keep` and `remove` parameters of the `kissplice2counts` function allow to indicate the type of splicing events to analyse or to remove from the analysis. These two parameters take a character vector indicating the types of events to keep or remove. The events names should come from this list: `deletion`, `insertion`, `IR`, `ES`, `altA`, `altD`, `altAD`, `alt`, `unclassified`. In the `remove` parameter, the event name `MULTI` can also be used. By default, all types of events are analysed.

If we want 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, counts = 2, pairedEnd = TRUE,
    k2rg = TRUE, keep = c("ES"), remove = c("MULTI"))
```

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 (see left panel of Figure 3)
- the factor map formed by the first two axes of a principal component analysis (PCA) (see right panel of Figure 3)

These two graphs show the similarities and the differences between the analyzed samples. We expect to cluster together the replicates of the same condition. If a sample is contaminated or has an abnormality that will influence the differential analysis, it will certainly not be clustered as expected. 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 this example, the plots are reassuring. Indeed, 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.

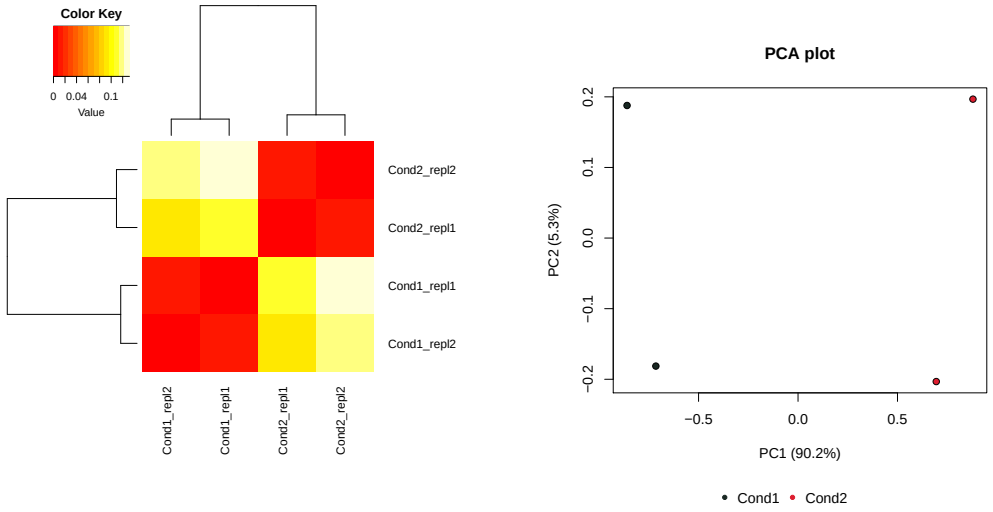

Figure 3: **Quality control plots.** Left: Heatmap of the sample-to-sample distances. Right: Principal Component Analysis.

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 the **R** working directory) or to the path where the user wants to store his/her graphs.

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)

[1] "Pre-processing the data..."
[1] "Trying to fit models on data..."
[1] "Searching for best model and computing pvalues..."
[1] "Computing size of the effect and last cutoffs..."
```

The `diffExpressedVariants` function has three other parameters to change the filters or the flags applied on data:

- `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) $> 5\%$ will be present in the final table. If you wish to change this threshold, you will need to recompute everything.
- `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.

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 containing p-values before and after correction for multiple testing. `resultFitNBglmModel` is a data frame containing the results of the fitting of models 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 respectively section 2.4.1 and section 2.4.2.

To control that the condition of application of the Benjamini Hochberg multiple testing correction procedure is fulfilled, 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, the distribution obtained is not representative. Representative histograms can be obtained with the two complete datasets presented in the case studies (section 4). As expected, the histograms show a uniform distribution with a peak near 0 (Figure 4).

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 is as following:

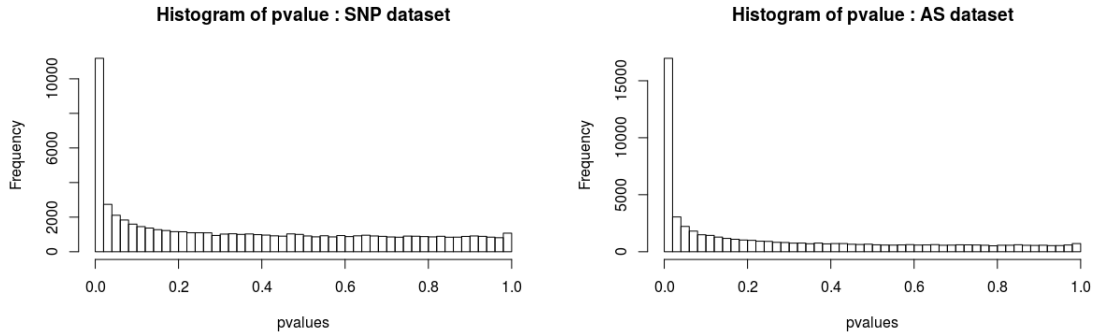

Figure 4: **Distribution of p-values before correction for multiple testing.** Left: for the complete dataset presented in section 4.1. Right: for the complete dataset presented in section 4.2.

ID	Length_diff	Variant1_condition_1_repl1_Norm
bcc_83285|Cycle_2	99	87
bcc_161433|Cycle_2	47	16
bcc_135201|Cycle_433392	104	80

Variant1_condition_1_repl2_Norm	Variant1_condition_2_repl1_Norm
46	37
17	68
52	33

Variant1_condition_2_repl2_Norm	Variant2_condition_1_repl1_Norm
39	2
62	47
35	2

Variant2_condition_1_repl2_Norm	Variant2_condition_2_repl1_Norm
5	113
33	8
1	33

Variant2_condition_2_repl2_Norm	Adjusted_pvalue	Deltaf/DeltaPSI	lowcounts
161	0	-0.7651	FALSE
1	0	0.6980	FALSE
59	0	-0.6956	FALSE

This table contains in columns the following information:

- `ID` is the event identifier. An 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 `FALSE` value means that the event has high counts (counts above 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 created using the `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 = "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 < 0.05 and the Deltaf/DeltaPSI absolute value > 0.10, the following command can be used:

```
> writeOutputKissDE(myResults, output = "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. Like previously explained (section 2.4.1), users can choose to save only events passing thresholds, in a file called `merge_K2RG_results_table.tab`:

```
> writeOutputKissDE(myResults_K2RG, output = "merge_K2RG_results_table.tab",
    adjPvalMax = 0.05, dPSImin = 0.1)
```

2.4.2 f/PSI table

The `f/psiTable` output of the function `diffExpressedVariants` contains the f values for SNV analysis and 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        0.00974026        0.01904359       0.005790264
2 bcc_108176|Cycle_0        0.03764306        0.06106575       0.038121280
3 bcc_120508|Cycle_0        0.94573955        0.94868871       0.961552545
  condition_2_repl2
1        0.01163492
2        0.03008335
3        0.94233937
```

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`:

```
> 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** (Love et al., 2014) package. The goal of this normalization step is to remove technical factors that could influence the differential analysis. By using **DESeq2** normalization, we correct for library size, because the sequencing depth can vary between samples.

3.2 Estimation of dispersion

A model to describe the counts distribution is first chosen. When working with biological replicates, 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, which is chosen in **kissDE**. However, due to numerical instabilities associated with the estimation of Negative Binomial

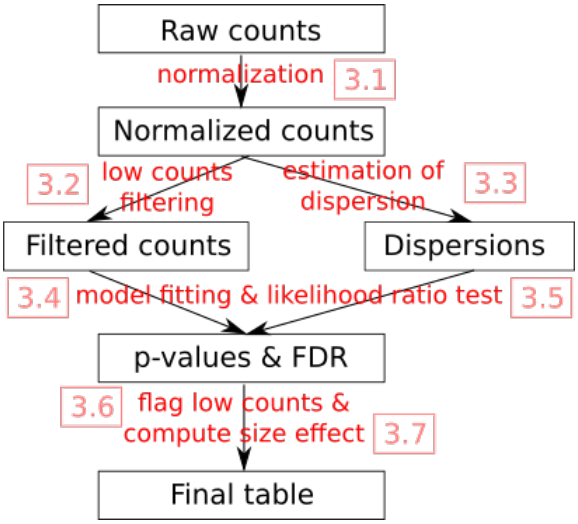

Figure 5: **The different steps of the `diffExpressedVariants` function. Numbers in light red point to the section of this vignette explaining the step.**

parameters, we use a model selection approach to determine which model is the best to handle the overdispersion parameter ϕ. We first estimate a model without overdispersion using the **glmnet R** package (Friedman et al., 2010) (model $\mathcal{M}(\phi = 0)$). We then consider two different estimation methods for the parameter ϕ, namely a global estimation approach using the **aod R** package (Lesnoff and Lancelot, 2012) (model $\mathcal{M}(\phi = \phi_{\text{global}})$), and a variant-specific parameter using the **DSS R** package (Wu et al., 2013; Feng et al., 2014; Wu et al., 2015; Park and Wu, 2016) (model $\mathcal{M}(\phi = \phi^i_{\text{DSS}})$. The bayesian information criterion (BIC) is used to choose the best model out of the three.

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.

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.

	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 en event filtered out before the differential analysis, because less than 10 reads support each variant.

3.4 Model

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 following:

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

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 $\mathcal{M}_\prime$ and $\mathcal{M}_\infty$, we perform a Likelihood Ratio Test (LRT). 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. The LRT is performed with one degree of freedom, which corresponds to the supplementary interaction parameter. p-values are then adjusted with a 5% false discovery rate (FDR) following a Benjamini-Hochberg procedure 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 classify this in event in `'Low_Counts'` column with a `TRUE` value. 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, 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 < 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:

$$PSI \;=\; f \;=\; \frac{\#counts\_variant_1}{\#counts\_variant_1 + \#counts\_variant_2}$$

$$\Delta PSI \;=\; PSI_{cond1} - PSI_{cond2}$$

$$\Delta f \;=\; f_{cond1} - f_{cond2}$$

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 is not computed.
- Then mean PSI/f are computed for each condition. If more than half of individual PSI/f were not calculated at the previous step, the mean PSI/f is not computed.
- Finally, we output ΔPSI/Δf. Unless one of the mean PSI/f of a condition could not be estimated, ΔPSI/Δf is calculated subtracting one condition PSI/f from another. This does not always follow the order of the conditions you entered in the first place. To know the order, you must report to the output file. ΔPSI/Δf are calculated subtracting the first condition appearing in the header from the one after. ΔPSI/Δf absolute value vary between 0 and 1, with values close to 0 indicating low effects and values close to 1 strong effects.

4 Case studies

4.1 kissDE on SNV data

RNA-Seq data can be used to study SNVs (SNPs, mutations, RNA editing) in expressed regions of the genome. Such SNVs can be detected and reported by **KisSplice**. In this first following example, an analysis of SNPs is done with **kissDE** on RNA-Seq data from the human GEUVADIS project (Lappalainen et al., 2013). The sample data presented here is a subset of the case study presented in Lopez-Maestre et al. (2016).

4.1.1 Dataset

The dataset used in this example comes from the human GEUVADIS project. Two populations was 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.

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

The output file of **KisSplice** is a fasta file that stores SNVs found in the dataset. Its structure is described in the section 2.1.3. Following, the four first lines of this fasta output file:

```
>bcc_44787|Cycle_833|Type_0a|upper_path_Length_83|C1_359|C2_369|C3_0|C4_0|
C5_0|C6_0|C7_0|C8_0|Q1_55|Q2_52|Q3_0|Q4_0|Q5_0|Q6_0|Q7_0|Q8_0|rank_0.00293
GAGTATCTGGTTGGCCTGGTATCAGCAGAAACCAGGGAAAGCCCCAAAACTCCTAATCCATAAGGCCTCTACTTTA
```

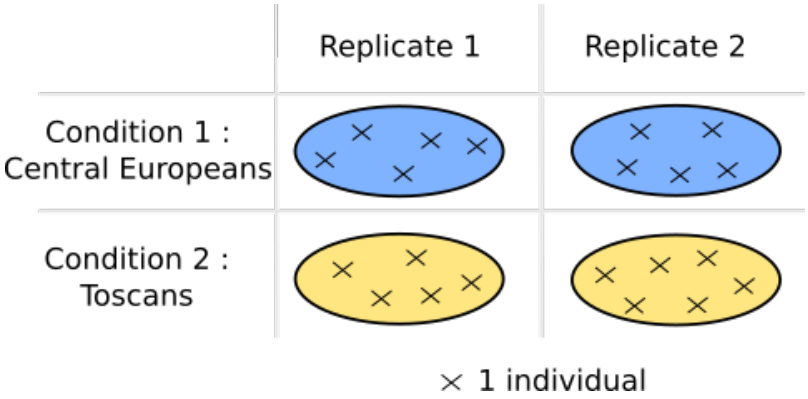

Figure 6: **Experimental design of the SNP dataset.**

```
GAAGCTG
>bcc_44787|Cycle_833|Type_0a|lower_path_Length_83|C1_42|C2_44|C3_0|C4_0|C5_0
|C6_0|C7_0|C8_0|Q1_61|Q2_62|Q3_0|Q4_0|Q5_0|Q6_0|Q7_0|Q8_0|rank_0.00293
GAGTATCTGGTTGGCCTGGTATCAGCAGAAACCAGGGAAAGACCCAAAACTCCTAATCCATAAGGCCTCTACTTTA
GAAGCTG
```

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 by their SNV content.

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.

4.1.2 Load data

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

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

```
> fileIn <- system.file("extdata", "output_kissplice_SNV.fa", package = "kissDE")
> myCounts_SNV <- kissplice2counts(fileIn, 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 that describes the experimental plan must also be provided. In the example, there are two replicates of TSC and two replicates of CEU. So the `myConditions_SNV` vector is defined:

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

4.1.3 Quality control

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

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

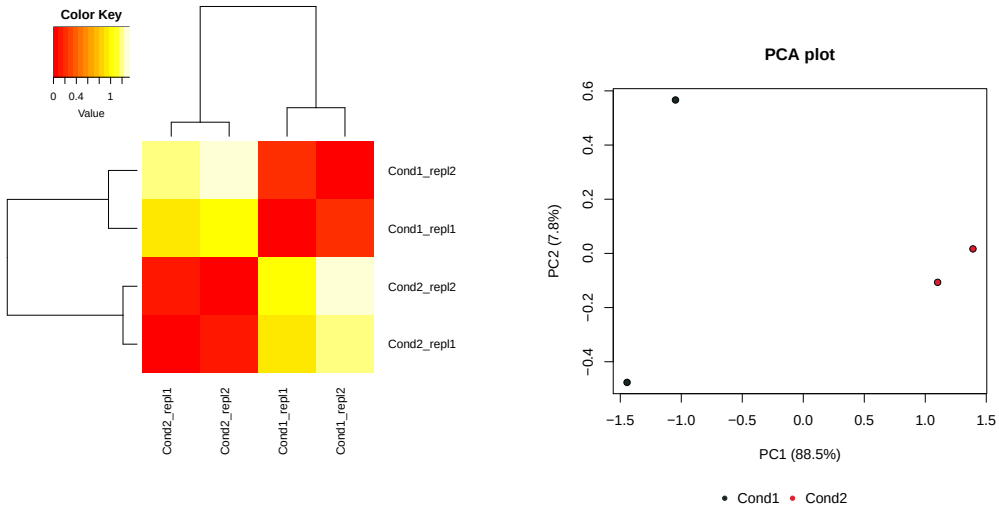

Figure 7: **Quality control plots on SNV data.** Left: Heatmap of the sample-to-sample distances on SNV data. Right: Principal Component Analysis on SNV data.

On both plots returned by the `qualityControl` function (Figure 7), 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 7), the samples of the same condition

cluster together. On the PCA plot (right of Figure 7), the first principal component (which resumes 88% of the total variance) clearly discriminate the samples of the two conditions.

4.1.4 Differential analysis

The main function of the **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 sequence of successive events is printed:

```
> myResult_SNV <- diffExpressedVariants(myCounts_SNV, myConditions_SNV)

[1] "Pre-processing the data..."
[1] "Trying to fit models on data..."
[1] "Searching for best model and computing pvalues..."
[1] "Computing size of the effect and last cutoffs..."

> head(myResult_SNV$finalTable, n = 3)
```

	ID	Length_diff	
bcc_44787|Cycle_320265	bcc_44787|Cycle_320265	0	
bcc_44787|Cycle_421687	bcc_44787|Cycle_421687	0	
bcc_61380|Cycle_1	bcc_61380|Cycle_1	0	
	Variant1_CEU_repl1_Norm	Variant1_CEU_repl2_Norm	
bcc_44787|Cycle_320265	1921	1387	
bcc_44787|Cycle_421687	4	85	
bcc_61380|Cycle_1	10	23	
	Variant1_TSC_repl1_Norm	Variant1_TSC_repl2_Norm	
bcc_44787|Cycle_320265	0	2	
bcc_44787|Cycle_421687	843	1342	
bcc_61380|Cycle_1	7	2	
	Variant2_CEU_repl1_Norm	Variant2_CEU_repl2_Norm	
bcc_44787|Cycle_320265	22	215	
bcc_44787|Cycle_421687	24	10450	
bcc_61380|Cycle_1	0	1	
	Variant2_TSC_repl1_Norm	Variant2_TSC_repl2_Norm	
bcc_44787|Cycle_320265	158	685	
bcc_44787|Cycle_421687	24	17	
bcc_61380|Cycle_1	2	20	
	Adjusted_pvalue	Deltaf/DeltaPSI	lowcounts
bcc_44787|Cycle_320265	4.975013e-09	-0.9258	FALSE
bcc_44787|Cycle_421687	2.736048e-04	0.9044	FALSE
bcc_61380|Cycle_1	5.118573e-03	-0.8883	FALSE

The first event in the `myResult_SNV` output has a low p-value (`Adjusted_pvalue` column, equal to `4.975013e-09`) and a very contrasted Δf (`Deltaf/DeltaPSI` column, equal to `-0.9258`) 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.

4.1.5 Export results

We consider as significant the events that have a p-value adjusted lower than 5%. Results passing this threshold are thus extracted from `myResults_SNV` thanks to the `adjPvalMax` parameter and are saved in a `final_table_significants.tab` file, thanks to the `writeOutputKissDE` function, as follows:

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

4.2 kissDE on alternative splicing data

This second 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 Benoit-Pilven et al.

4.2.1 Dataset

The data used in this example of AS comes from the ENCODE project (Djebali et al., 2012). The chosen 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.

KisSplice has been used to analyse these two conditions. Then, the *type 1* results from **KisSplice** were mapped to the reference genome with the **STAR** (Dobin et al., 2013) software and analyzed with **KisSplice2refgenome**. **KisSplice2refgenome** enables to annotate the AS events discovered by **KisSplice**. It assigns each event to the gene it maps to and to a type of alternative splicing event: Exon Skipping (ES), Intron Retention (IR), Alternatid Donor (AltD), Alternative Acceptor (AltA), ... For any question 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. Following, is an extract of this file (the first 3 rows and first 11 columns), where each row is one alternative splicing event:

```
        Gene_Id Gene_name Chromosome_and_genomic_position Strand Event_type
ENSG00000163875     MEAF6          chr1:37962165-37967445      -         ES
ENSG00000117620   SLC35A3        chr1:100435679-100459133      +         ES
ENSG00000125814      NAPB         chr20:23375598-23383670      -         ES
Variable_part_length Frameshift_? CDS_?   Gene_biotype
                  30          No   Yes protein_coding
                  99          No   Yes protein_coding
                  47         Yes   Yes protein_coding
number_of_known_splice_sites/number_of_SNPs
              all_splice_sites_known_(4_ss)
              all_splice_sites_known_(4_ss)
              all_splice_sites_known_(4_ss)
```

4.2.2 Load data

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

This conversion in a table of counts is stored in a `myCounts_AS` object (for a detailed description of its structure, see section 2.1.4) and can be done as follows:

```
> 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_68965|Cycle_4            30       2       1      23       8
2  bcc_68965|Cycle_4            82      33      14       6       3
3  bcc_83285|Cycle_2            99     105      41      15      26
4  bcc_83285|Cycle_2            81       2       5     100     150
5 bcc_161433|Cycle_2            47      20      17      60      58
6 bcc_161433|Cycle_2            80      58      33       7       1
```

To perform the differential analysis, a vector that describes the experimental plan must also be provided. In the example, there are two replicates of the SK-N-SH cell line without

treatment (SKNSH) and two replicates of the same cell line treated with retinoic acid (SKSNH-RA). So the `myConditions_AS` vector is defined:

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

4.2.3 Quality control

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

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

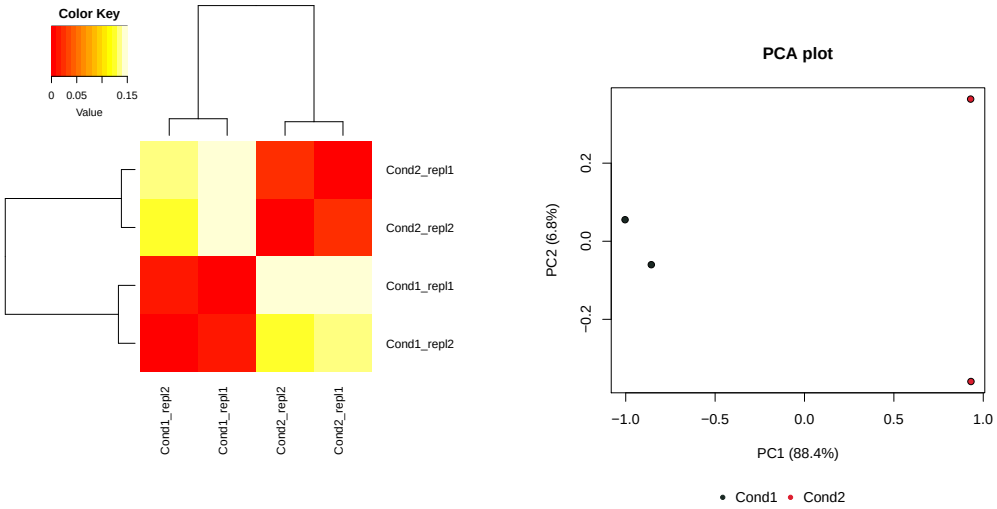

Figure 8: **Quality control plots on alternative data.** Left: Heatmap of the sample-to-sample distances for the alternative splicing dataset. Right: Principal Component Analysis for the alternative splicing dataset.

On both plots returned by the `qualityControl` function (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 resumes 88% of the total variance) clearly discriminates the samples of the two conditions.

4.2.4 Differential analysis

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

```
> myResult_AS <- diffExpressedVariants(myCounts_AS, myConditions_AS)

[1] "Pre-processing the data..."
[1] "Trying to fit models on data..."
[1] "Searching for best model and computing pvalues..."
[1] "Computing size of the effect and last cutoffs..."

> head(myResult_AS$finalTable, n = 3)

                                        ID Length_diff
bcc_83285|Cycle_2             bcc_83285|Cycle_2          18
bcc_52250|Cycle_0             bcc_52250|Cycle_0          82
bcc_135201|Cycle_433392 bcc_135201|Cycle_433392          22
                        Variant1_SKNSH_repl1_Norm Variant1_SKNSH_repl2_Norm
bcc_83285|Cycle_2                              83                        41
bcc_52250|Cycle_0                              10                        22
bcc_135201|Cycle_433392                        40                        27
                        Variant1_SKNSH-RA_repl1_Norm
bcc_83285|Cycle_2                                 17
bcc_52250|Cycle_0                                 16
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                           113
bcc_52250|Cycle_0                               0                            19
bcc_135201|Cycle_433392                         1                            33
                        Variant2_SKNSH-RA_repl2_Norm Adjusted_pvalue
bcc_83285|Cycle_2                                162    0.000000e+00
bcc_52250|Cycle_0                                 24    1.886453e-07
bcc_135201|Cycle_433392                           59    5.701772e-12
                        Deltaf/DeltaPSI lowcounts
bcc_83285|Cycle_2               -0.8042     FALSE
bcc_52250|Cycle_0               -0.7389     FALSE
bcc_135201|Cycle_433392         -0.7152     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 `-0.8042`) 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, there seems to be a switch of minor/major isoform.

4.2.5 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)
```

4.3 Time / Requirements

The data conversion with the `kissplice2counts` function and the statistical analysis done with the `diffExpressedVariants` function can take some time to run. Here is an example of the running time of this 2 functions 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	
			`kissplice2counts`	`diffExpressedVariants`
AS data	`counts`=2, `pairedEnd`=TRUE	68497	14m	2h 18m
SNV data	`counts`=0, `pairedEnd`=TRUE	64824	6m	2h 10m

Table 3: Running time of the two principal functions of **kissDE**(`kissplice2counts` and `diffExpressedVariants`) for two datasets (AS dataset from the ENCODE project (Djebali et al., 2012) described in section 4.2 and SNP dataset from the GEUVADIS project (Lappalainen et al., 2013) described in section 4.1).

5 Session info

Aurélie

References

S. Djebali, C. A. Davis, A. Merkel, A. Dobin, T. Lassmann, A. Mortazavi, A. Tanzer, J. Lagarde, W. Lin, F. Schlesinger, C. Xue, G. K. Marinov, J. Khatun, B. A. Williams, C. Zaleski, J. Rozowsky, M. Roder, F. Kokocinski, R. F. Abdelhamid, T. Alioto, I. Antoshechkin, M. T. Baer, N. S. Bar, P. Batut, K. Bell, I. Bell, S. Chakrabortty, X. Chen, J. Chrast, J. Curado, T. Derrien, J. Drenkow, E. Dumais, J. Dumais, R. Duttagupta,

E. Falconnet, M. Fastuca, K. Fejes-Toth, P. Ferreira, S. Foissac, M. J. Fullwood, H. Gao, D. Gonzalez, A. Gordon, H. Gunawardena, C. Howald, S. Jha, R. Johnson, P. Kapranov, B. King, C. Kingswood, O. J. Luo, E. Park, K. Persaud, J. B. Preall, P. Ribeca, B. Risk, D. Robyr, M. Sammeth, L. Schaffer, L.-H. See, A. Shahab, J. Skancke, A. M. Suzuki, H. Takahashi, H. Tilgner, D. Trout, N. Walters, H. Wang, J. Wrobel, Y. Yu, X. Ruan, Y. Hayashizaki, J. Harrow, M. Gerstein, T. Hubbard, A. Reymond, S. E. Antonarakis, G. Hannon, M. C. Giddings, Y. Ruan, B. Wold, P. Carninci, R. Guigo, and T. R. Gingeras. Landscape of transcription in human cells. *Nature*, 489(7414):101–108, 2012. ISSN 0028-0836. doi: 10.1038/nature11233.

A. Dobin, C. A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. R. Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics (Oxford, England)*, 29(1):15–21, jan 2013. ISSN 1367-4811. doi: 10.1093/bioinformatics/bts635.

H. Feng, K. N. Conneely, and H. 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.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Softwarey*, 33(1):1–22, 2010. ISSN 1548-7660. doi: 10.18637/jss.v033.i01.

T. Lappalainen, M. Sammeth, M. R. Friedländer, P. A. C. 't Hoen, J. Monlong, M. A. Rivas, M. Gonzàlez-Porta, N. Kurbatova, T. Griebel, P. G. Ferreira, M. Barann, T. Wieland, L. Greger, M. van Iterson, J. Almlöf, P. Ribeca, I. Pulyakhina, D. Esser, T. Giger, A. Tikhonov, M. Sultan, G. Bertier, D. G. MacArthur, M. Lek, E. Lizano, H. P. J. Buermans, I. Padioleau, T. Schwarzmayr, O. Karlberg, H. Ongen, H. Kilpinen, S. Beltran, M. Gut, K. Kahlem, V. Amstislavskiy, O. Stegle, M. Pirinen, S. B. Montgomery, P. Donnelly, M. I. McCarthy, P. Flicek, T. M. Strom, T. G. Geuvadis Consortium, H. Lehrach, S. Schreiber, R. Sudbrak, A. Carracedo, S. E. Antonarakis, R. Häsler, A.-C. Syvänen, G.-J. van Ommen, A. Brazma, T. Meitinger, P. Rosenstiel, R. Guigó, I. G. Gut, X. Estivill, and E. T. Dermitzakis. Transcriptome and genome sequencing uncovers functional variation in humans. *Nature*, 501(7468):506–11, 2013. ISSN 1476-4687. doi: 10.1038/nature12531.

M. Lesnoff and R. Lancelot. *aod: Analysis of Overdispersed Data*, 2012. URL http://cran.r-project.org/package=aod. R package version 1.3.

H. Lopez-Maestre, L. Brinza, C. Marchet, J. Kielbassa, S. Bastien, M. Boutigny, D. Monnin, A. E. Filali, C. M. Carareto, C. Vieira, F. Picard, N. Kremer, F. Vavre, M.-F. Sagot, and V. 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. ISSN 0305-1048. doi: 10.1093/nar/gkw655.

M. I. Love, W. Huber, and S. Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome Biology*, 15(12):550, 2014. ISSN 1474-760X. doi: 10.1186/s13059-014-0550-8.

Y. Park and H. Wu. Differential methylation analysis for bs-seq data under general experimental design. *Bioinformatics*, 32(10):1446, 2016. doi: 10.1093/bioinformatics/btw026.

G. A. T. Sacomoto, J. Kielbassa, R. Chikhi, R. Uricaru, P. Antoniou, M.-F. Sagot, P. Peterlongo, and V. Lacroix. Kissplice: de-novo calling alternative splicing events from rna-seq data. *BMC bioinformatics*, 13(6):S5, 2012. ISSN 1471-2105. doi: 10.1186/1471-2105-13-S6-S5. URL http://dx.doi.org/10.1186/1471-2105-13-S6-S5.

H. Wu, C. Wang, and Z. Wu. A new shrinkage estimator for dispersion improves differential expression detection in rna-seq data. *Biostatistics*, 14(2):232, 2013. doi: 10.1093/biostatistics/kxs033.

H. Wu, T. Xu, H. Feng, L. Chen, B. Li, B. Yao, Z. Qin, P. Jin, and K. 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.