Supplementary file supporting the manuscript **A benchmarking of workflows for detecting differential splicing and differential expression at isoform level in human RNA-seq studies**

1- Tools and workflows for detecting differential isoform expression and differential splicing

A total of nine workflows for RNA-seq analysis at isoform level were studied and compared. Five of those were used to evaluate the occurrence of *Differential Isoform Expression* (DIE), whereas the other four pipelines were used to analyze *Differential Splicing* (DS). Both, alignment against genome and transcriptome reference were required. Pipelines for DIE analysis used expression counts at the isoform level; while exon-level counts were used to perform DS study. The Supplementary Table S1 describes the workflows used here.

Supplementary Table S1: Description of the evaluated workflows for differential expression analysis at isoform and splicing level.

Workflow	DE type	Aligner	Quantifier	DE tool
EBSeq	DIE	Bowtie (v1.0.0)	RSEM (v.1.2.30)	EBSeq (v1.10.0)
DESeq2				DESeq2 (v1.10.1)
NOISeq				NOISeq (v2.14.1)
Limma/LimmaDS	DIE/DS			Voom-Limma (v3.26.9)
		Tophat2 (v2.0.9)	DEXSeq python script (v. 1.16.10)	
Cufflinks/CufflinksDS			Cufflinks (v2.1.1)	Cuffdiff (v2.1.1)
DEXSeq	DS		DEXSeq python script (v. 1.16.10)	DEXSeq (v. 1.16.10)
SplicingCompass			coverageBed (v2.17.0)	SplicingCompass (v1.0.1)

DE, Differential Expression; DIE, Differential Isoform Expression; DS, Differential Splicing

2- Simulation study

The study consisted of two main parts, the Simulation and the Differential Expression Analysis. The simulation was based on a typical case-control RNA-seq experiment. We chose a study having 30 samples (GSE22260) involving ten matched samples from Normal (C) and Tumoral (T) tissue and ten Tumoral samples from the remaining ten patients. We decided to conserve 20 non-matched samples, ten from Normal tissue and ten from non-matched Tumor tissue to avoid samples correlation. By means of a quality control procedure, we determined and then filtered out four outlier samples. Thus, a total of 16 samples, eight replicates per each C and T conditions, were considered for the simulation. After that, genome and transcriptome indexes from the human genome/transcriptome fasta and gtf files (downloaded from ENSEMBL, hg19v37) were generated. Using the 16 fastq files from the real data set, we estimated the `RSEM` model parameters and transcript expression levels, for each sample. Then, the expression matrices were loaded into R in order to simulate expression profiles controlling DIE and DS. The negative binomial distribution was used to obtain the replicated isoform counts for each condition. Sample mean and variance were obtained in two steps. First, these were estimated from the real expression matrix. Then, a set of genes was randomly chosen and their estimated parameters were modified to simulate differential expression. In this step, the simulation study was divided into four subgroups (see also Supplementary Table S2):

DIE: Differential Isoform Expression, involved genes with absolute expression changes in all their isoforms between C and T, but without changes in isoform proportions. The simulated DIE changes were represented by fold changes of 2, 3, 4 and 5 in both directions; it means that for instance, some genes had their isoforms with the expression increased two times in condition T with respect to their expression in C; whereas others genes had their isoforms with the expression increased two times in condition C in comparison with their expression in T. Let a gene g, n_g (>1) denoted the number of isoforms of this gene. If g was selected to be simulated as DIE, it means that each isoform (i) of g will have a fold change a, equal to 2, 3, 4 or 5, for the TvsC comparison. We simulated the mean and variance for each isoform in condition C and T following Eq. S1:

$$\begin{array}{c} \mu_{Ci} = \mu_{C0i} \ \ and \ \ \sigma_{Ci}^2 = \sigma_{C0i}^2 \quad \forall i = 1 \ldots n_g \\ \mu_{Ti} = a\mu_{C0i} \ \ and \ \ \sigma_{Ci}^2 = a^2\sigma_{C0i}^2 \quad \forall i = 1 \ldots n_g \end{array} \quad \text{Eq S1.}$$

where μ_{C0i} and σ^2_{C0i} are the sample mean and variances, respectively, observed for the isoform *i* in the original expression matrix for condition C. On the opposite, fold changes of 0.5, 0.33, 0.25 and 0.2 corresponded to folds of 2, 3, 4 and 5 for the comparison CvsT. Consequently, these changes were simulated increasing the mean and variances of C, following equation Eq. S2:

$$\begin{array}{c} \mu_{Ci} = a\mu_{C0i} \ \ and \ \ \sigma_{Ci}^2 = a^2\sigma_{C0i}^2 \quad \forall i = 1 \ldots n_g \\ \mu_{Ti} = \mu_{C0i} \ \ and \ \ \sigma_{Ci}^2 = \sigma_{C0i}^2 \quad \forall i = 1 \ldots n_g \end{array} \quad \text{Eq S2.}$$

DE: Differential Expression, included genes with only one annotated transcript changing its expression between C and T. The considered fold changes were 2 and 4, in both directions. These changes were simulated using Eq. S1 and Eq. S2, with the particularly that n_g=1.

DS: Differential Splicing, involved genes with more than two annotated isoforms changing their proportions between C and T, without changes in the overall gene expression. The major isoform (M) proportion was changed from 0 in condition C to 0.7 in condition T (0-0.7), 0.1-0.4, 0.3-0.6 and 0.5-0.8 in both directions; while the rest of the isoforms (n_g-1) were simulated to have the same relative expression, equal to the remaining proportion (1-p) divided for n_g-1. For instance, a gene with three isoforms had its major isoform with a proportion of 0.5 in condition C (p_C) and of 0.8 in T (p_T). The other two isoforms had proportions of 0.25 and 0.1, in conditions C and T, respectively. To simulate the described changes, we used equation Eq. S3 for the estimation of major isoform parameters and Eq. S4 for the rest of the gene isoforms.

$$\begin{array}{c} \mu_{CM} = p_C\mu_{Cg} \ \ and \ \ \sigma_{CM}^2 = p_C^2\sigma_{Cg}^2 \\ \mu_{TM} = p_T\mu_{Cg} \ \ and \ \ \sigma_{TM}^2 = p_T^2\sigma_{Cg}^2 \end{array} \quad \text{Eq S3.}$$

$$\mu_{Ci} = p_{Ci}\mu_{Cg} \text{ and } \sigma^2_{Ci} = p^2_{Ci}\sigma^2_{Cg}\,, \text{ with } p_{Ci} = \frac{1-p_C}{n_g-1} \quad \forall i = 1 \dots n_g \wedge i \neq M$$
$$\mu_{Ti} = p_{Ti}\mu_{Cg} \text{ and } \sigma^2_{Ti} = p^2_{Ti}\sigma^2_{Cg}\,, \text{ with } p_{Ti} = \frac{1-p_T}{n_g-1} \quad \forall i = 1 \dots n_g \wedge i \neq M \qquad \text{Eq S4.}$$

Since human genes have different numbers of isoforms, from one to more than 20, genes were grouped according to their number of isoform and this grouping was considered when DS subgroups were assigned in order to avoid non-biological situations. For instance, we observed that major isoform proportions close to 0.7 were mainly for genes having at most four annotated transcripts; thus, the simulation of 0-0.7 DS case only considered genes fulfilling this.

DIEDS: Differential Isoform Expression and Differential Splicing, genes with two or more transcripts with changes in isoform proportion and in the overall gene expression between C and T conditions. In particular, we simulated the following cases: 0.5-0.8-0.5, 2-0.8-0.5, 4-0.8-0.5, 2-0.8-0.3 and 4-0.8-0.3. Each case has three ordered numbers indicating the fold change (a) TvsC, the proportion of major isoform in C (p_C) and in T (p_T), respectively. To simulate the described changes, we used equation Eq. S5 for the major isoform changes and Eq. S6 for the rest of the gene isoforms.

$$\mu_{CM} = p_C\mu_{Cg} \text{ and } \sigma^2_{CM} = p^2_C\sigma^2_{Cg}$$
$$\mu_{TM} = ap_T\mu_{Cg} \text{ and } \sigma^2_{TM} = a^2p^2_T\sigma^2_{Cg} \qquad \text{Eq S5.}$$

$$\mu_{Ci} = p_{Ci}\mu_{Cg} \text{ and } \sigma^2_{Ci} = p^2_{Ci}\sigma^2_{Cg}\,, \text{ with } p_{Ci} = \frac{1-p_C}{n_g-1}, \quad \forall i = 1 \dots n_g \wedge i \neq M$$
$$\mu_{Ti} = ap_{Ti}\mu_{Cg} \text{ and } \sigma^2_{Ti} = a^2p^2_{Ti}\sigma^2_{Cg}\,, \text{ with } p_{Ti} = \frac{1-p_T}{n_g-1}, \quad \forall i = 1 \dots n_g \wedge i \neq M \qquad \text{Eq S6.}$$

For genes simulated as not differentially expressed, we set mean and variance parameters of condition T equal to the parameters observed for condition C. Once all mean and variance parameters for both conditions were defined, we randomly generated negative binomial isoform counts to simulate replication. Finally, the simulated expression profiles were used to call RSEM obtaining simulated paired-end short reads for each sample in each simulated scenario. In order to provide statistical significance, we perform ten simulations for each experimental scenario (S1, S2 and S3).

3- Differential expression analysis.

For the analysis part, we started by preparing annotation files and index files necessary for the subsequent steps. We used `Bowtie` to generate index file needed to perform the alignments against the reference transcriptome and genome. The annotation file (gff) was flattened using the `DEXSeq` python script `dexseq_prepare_annotation.py` (with --aggregate='no') and an `R` script provided by `SplicingCompass` developers. We aligned reads against the reference transcriptome, using `Bowtie` and then we performed the isoforms quantification by means of the `rsem-calculate-expression` script from `RSEM`. `Tophat2` and `Bowtie` were used to align reads against the genome reference. Genome alignments and the flattened annotation files were used to bins quantification with the `dexseq_count.py script` and the `coverageBed` tool for

DEXSeq and SplicingCompass, respectively. Finally, the genome alignments were processed by Cufflinks.

For DE analysis, EBSeq, DESeq2, Limma and NOISeq over the raw counts at isoform level, generated by RSEM, were run. The expression matrices were pre-processed to filter out those isoforms having mean expression value less than four counts per million (cpm) in at least one condition (C or T) and those non-simulated isoforms. For DESeq2, we used the FDR adjusted p-value obtained using the nbinomWaldTest method. In the case of Limma, we first used the voom transformation to model the mean–variance relationship and then, we used the eBayes method to test differential expression obtaining the corresponding p-values. Those p-values were adjusted using the p.adjust R method with the "fdr" option. For EBSEq and NOISeq, the posterior probability of being differentially expressed (PPDE) was used to obtain the list of differential isoforms. Following the recommendations found in vignette packages, we used 1-PPDE as an equivalent to the FDR. In all cases, we used a significance threshold α = 0.05.

DS methods were run over the corresponding expression matrices. For SplicingCompass the initSigGenesFromResults method was used to obtain DS genes. DEXSeq and Limma were run over the expression matrices obtained using the DEXSeq python script following the package suggestions. In the case of Limma, exons having more than one cpm in at least three samples were kept to perform differential analysis. The voom transformation, lmFit and diffSplice methods were used to DS evaluation. For DEXSeq, we used the testForDEU and perGeneQValue methods to obtain an FDR adjusted p-value at gene level. In all cases, DS was evaluated considering a significance threshold α =0.05 for gene adjusted p-values.

Finally, Cuffdiff was run generating the isoform_exp.diff and splicing.diff files for DIE and DS respectively. Differential expression was analyzed using the same significance threshold above.

4- Evaluation of workflows performances

The pipelines results in all the simulations of each scenario (S1, S2 and S3) were evaluated using well-known performance measures. Detected differentially expressed isoforms and differentially spliced genes were classified as:

- True Positive (TP): Isoforms/genes simulated as differentially expressed showing adjusted p-values lower than 0.05.
- True Negative (TN): Isoforms/genes simulated without expression changes with adjusted p-values higher than 0.05.
- False Positive (FP): Isoforms/genes simulated without expression changes showing adjusted p-values lower than 0.05
- False Negative (FN): Isoforms/genes simulated as differentially expressed with adjusted p-values higher than 0.05.

In the case of DIE methods, those isoforms simulated as DIE or DS were considered as TP, since DS could cause DIE; but, to the FN definition, only isoforms simulated as DIE were considered because we did not control if the simulated DS changes caused DIE. In the case of DS method, those genes simulated to have changes in the proportion of their isoforms were considered (DS and DIEDS simulation groups). Then, several performance measures were computed:

- Accuracy: Proportion of true results (TP and TN) among the total number of examined cases.
- Sensitivity: So-called true positive rate (TPR) is the proportion of simulated DE isoforms (genes) correctly detected as such.
- False positive rate (FPR): is the proportion of simulated non-DE isoforms (genes) correctly detected as such.
- Precision: Or positive predicted value (PPV) is the proportion of isoforms (genes) detected as DE that were simulated as DE. In addition, the False Discovery Rate (FDR) is the complement of precision.
- F-score: Is the harmonic mean of precision and sensitivity.

Supplementary Table S3: Performance measures used to compare the evaluated methods. In the context of differential expression results, exist two groups: differentially expressed and not differentially expressed. Thus, an isoform/gene was a *positive* if its classification was the one simulated and a *negative* if it was incorrectly classified.

Measure	**Formula**
Accuracy, ACC	$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$
Sensitivity, TPR	$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$
False Positive Rate, FPR	$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}$
Precision, PPV	$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$
False Discovery Rate, FDR	$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{TP} + \mathrm{FP}} = 1 - \mathrm{PPV}$
F-score	$\mathrm{F} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$

TP, true positives; TN, true negatives; FP, false positives; FN, false negatives.

5- Overall differential expression results

Supplementary Table S4: Overall differential isoform expression results. The differentially expressed isoforms (DI) detected using the five workflows were obtained for the three simulated scenarios (S1, S2 and S3). The first row represents the average of the number of DI detected along the ten simulations of S1, S2 and S3. The next two rows contain the number of DI found in at least one simulation (*All DI*) and the percentage of those found in all the simulations (*%concordant DI*). The fourth and fifth rows show the amount of true DI (*TP*) found in at least one simulation and the percentage of those found in all the simulations (*%concordant TP*), respectively. Finally, the last two rows exhibit the amount of FP identified in at least one simulation and the percentage of those found in all the simulations (*%concordant FP*).

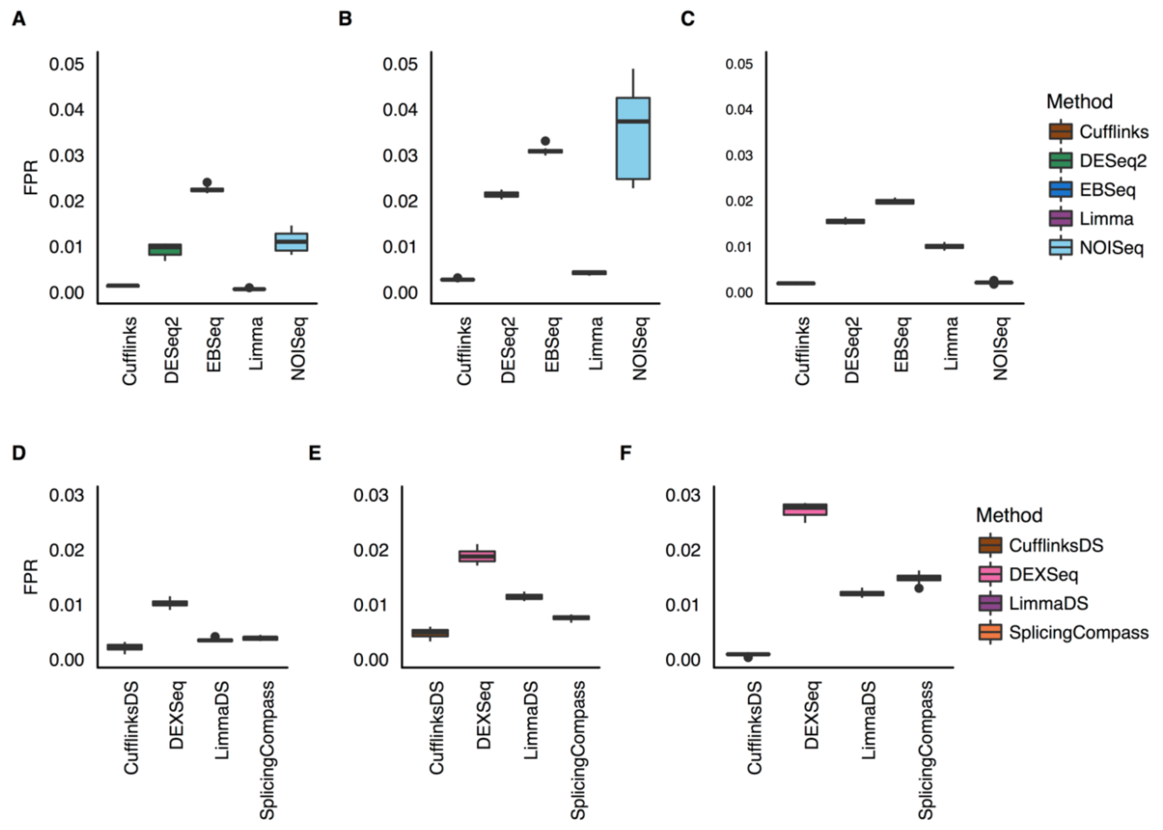
Variable	**EBSeq**			**DESeq2**			**NOISeq**			**Cufflinks**			**Limma**		
	S1	*S2*	*S3*	*S1*	*S2*	*S3*	*S1*	*S2*	*S3*	*S1*	*S2*	*S3*	*S1*	*S2*	*S3*
Mean DI	2679	6086	6305	1948	5577	6054	1961	6388	2882	414	1257	1030	*838*	3077	5238
All DI	8852	16553	13875	4937	13179	11916	4996	15956	4908	845	2460	1860	*1468*	5451	9927
%concordant DI	10.8	14.6	17.5	17.7	21.1	45.6	19	19.7	32.4	25.2	27.2	29.9	*29.2*	31.05	28.3
TP	3155	8037	8155	2739	7483	7770	2556	7742	4342	584	1816	1452	*1351*	4590	7177
%concordant TP	29.6	29.7	39.4	31.1	36.5	44.7	36.6	40	36.4	35.8	36.3	37.7	*31.4*	36.2	38.5
FP	5697	8516	5720	2198	5696	4146	2440	8214	566	261	644	408	*117*	861	2750
%concordant FP	0.4	0.5	0.7	1	1.1	1.4	0.7	0.9	1.8	1.5	1.6	1.5	*4.3*	3.8	1.6

Supplementary Table S5: Overall differential splicing results. The Alternative Spliced Genes (ASG) detected, using the evaluated four pipelines were obtained for the three simulated scenarios (S1, S2 and S3). The first row represents the mean of ASG detected along the ten simulations of each scenario. The next two rows contain the number of ASG found in at least one simulation (*All ASG*) and the percentage of those found in all the simulations (*%concordant ASG*), for each scenario. The fourth and fifth rows show the amount of true ASG found in at least one simulation (*TP*) and the percentage of those found in all the simulations (*%concordant TP*), respectively. Finally, the last two rows exhibit the amount of false ASG identified in at least one simulation (*FP*) and the percentage of those found in all the simulations (*%concordant FP*).

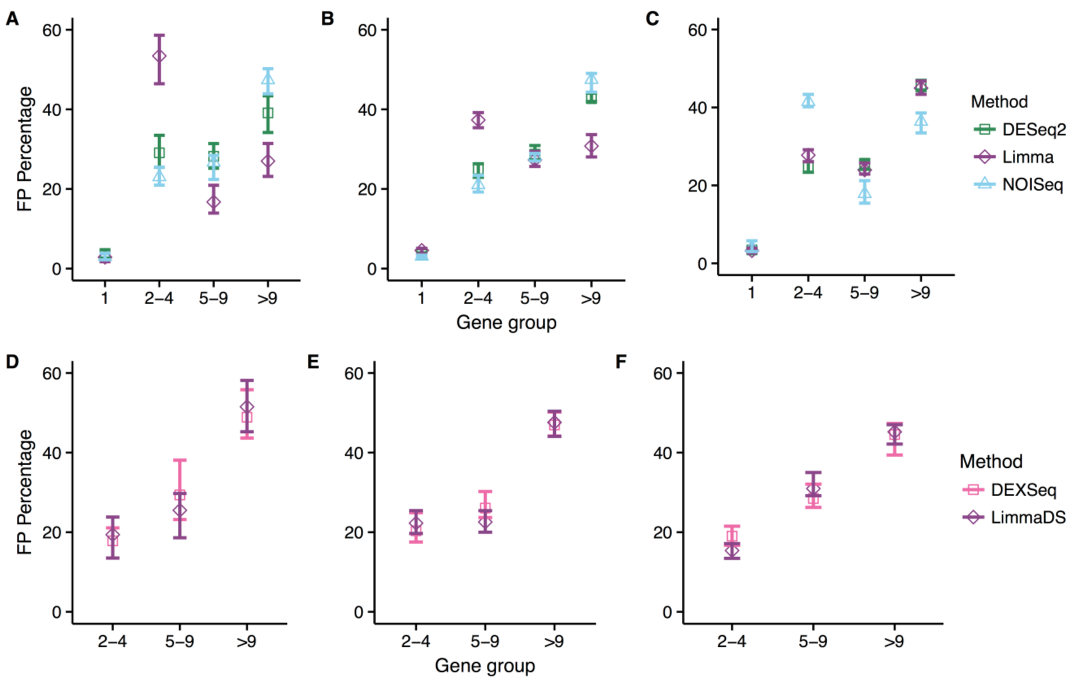
Variable	**CufflinksDS**			**DEXSeq**			**LimmaDS**			**SplicingCompass**		
	S1	*S2*	*S3*	*S1*	*S2*	*S3*	*S1*	*S2*	*S3*	*S1*	*S2*	*S3*
Mean ASG	106	303	226	423	913	1147	233	615	758	149	349	600
All ASG	318	779	468	1134	2027	2630	431	1060	1290	536	923	1256
%concordant ASG	11.3	13	17.3	16.4	22.9	29.6	26.2	28.6	30.7	6.2	9.5	17.5
TP	185	527	437	372	868	908	292	723	842	234	557	714
%concordant TP	19.5	19.4	18.8	44.9	46.8	58.7	34.2	37.6	41.4	11.5	13.5	24.6
FP	133	252	31	762	1159	1722	139	337	448	302	366	542
% concordant FP	0	0.4	3.2	2.5	5	3.4	9.4	15.7	10.5	2	1.6	8

Supplementary Table S6: Overall performance measures for the evaluated workflows. Each cell contains, for the indicated measurement, the average of the ten simulations performed for each scenario.

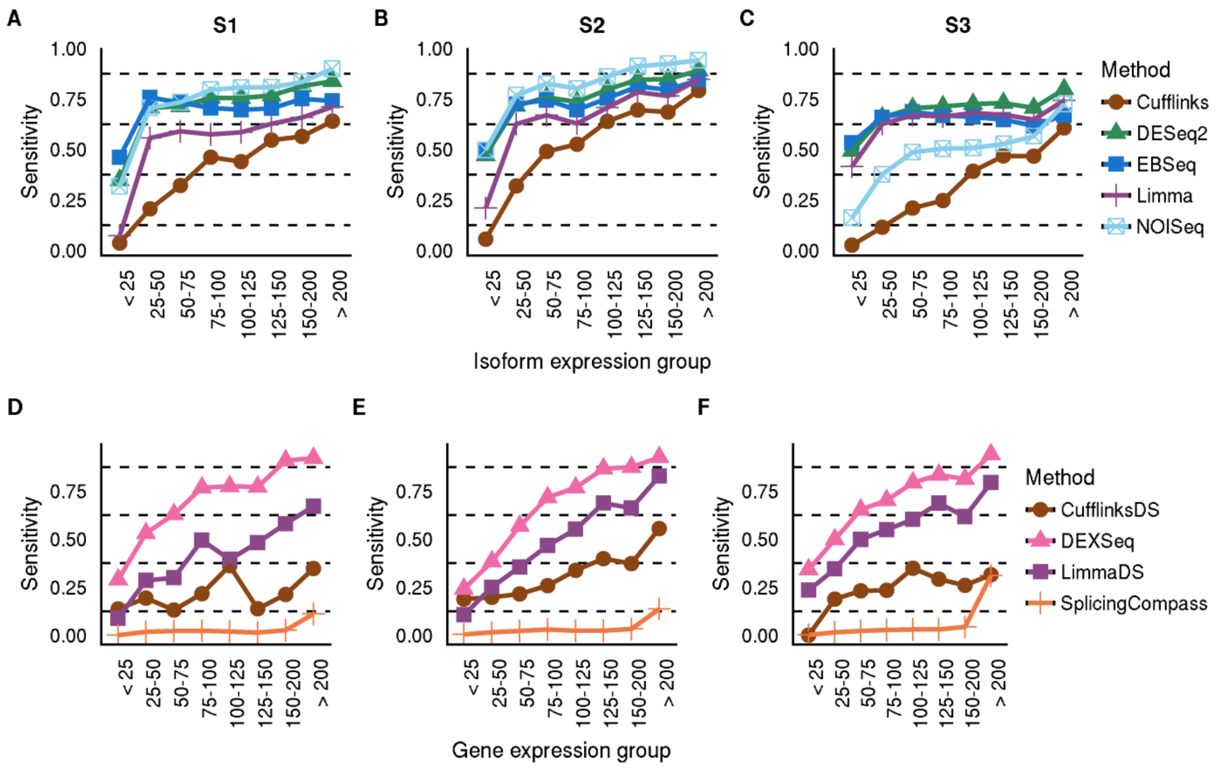
	Variable	Accuracy			Sensitivity			Precision			F-score		
		S1	*S2*	*S3*	*S1*	*S2*	*S3*	*S1*	*S2*	*S3*	*S1*	*S2*	*S3*
DIE	*EBSeq*	0.930	0.882	0.899	0.546	0.565	0.585	0.714	0.806	0.863	0.706	0.762	0.808
	DESeq2	0.918	0.875	0.885	0.468	0.550	0.580	0.842	0.856	0.891	0.715	0.777	0.817
	NOISeq	0.923	0.874	0.870	0.454	0.585	0.300	0.811	0.798	0.968	0.708	0.777	0.591
	Cufflinks	0.940	0.873	0.868	0.141	0.193	0.161	0.883	0.927	0.939	0.338	0.430	0.373
	Limma	0.926	0.868	0.897	0.233	0.336	0.516	0.973	0.947	0.917	0.484	0.610	0.781
DS	*SplicingCompass*	0.931	0.900	0.908	0.258	0.289	0.482	0.688	0.754	0.725	0.376	0.419	0.579
	DEXSeq	0.981	0.963	0.963	0.682	0.693	0.823	0.660	0.714	0.671	0.671	0.703	0.739
	CufflinksDS	0.962	0.922	0.919	0.270	0.347	0.289	0.867	0.903	0.977	0.410	0.501	0.446
	LimmaDS	0.980	0.953	0.965	0.471	0.528	0.669	0.819	0.794	0.814	0.598	0.634	0.734



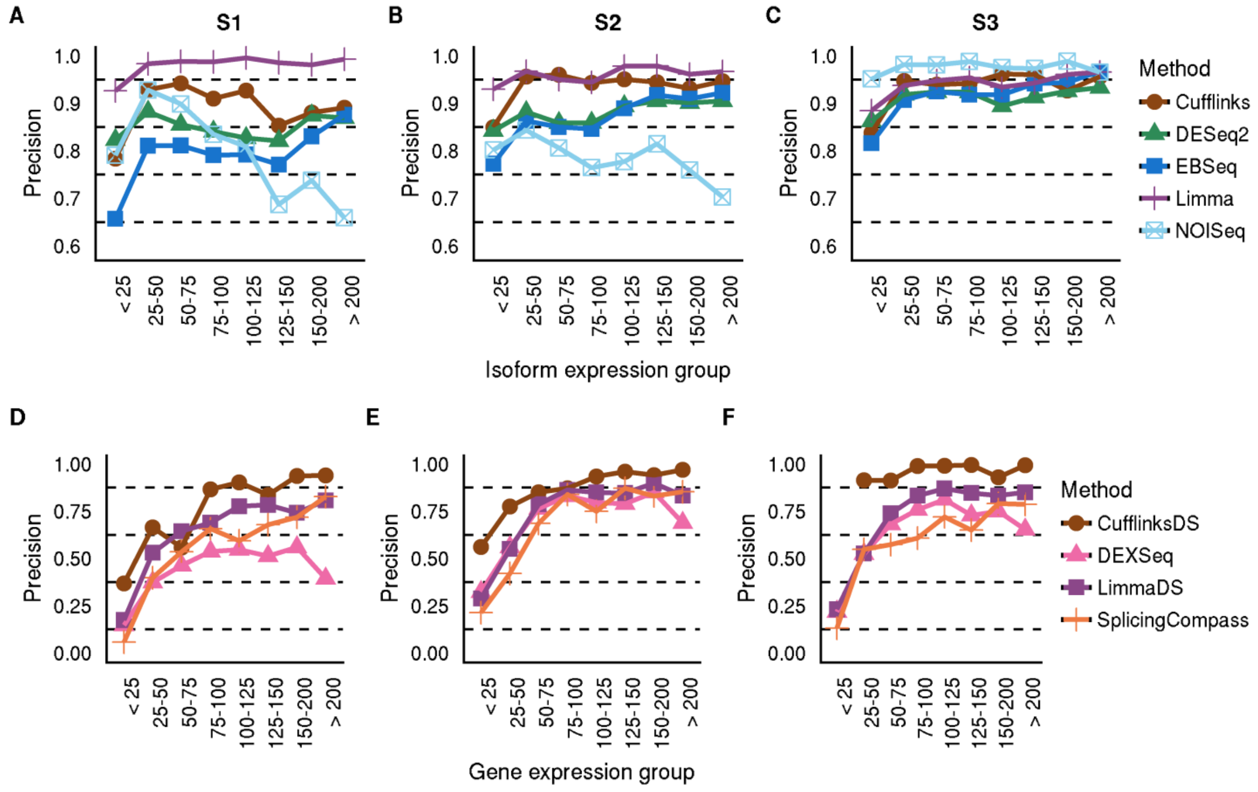
Supplementary Figure S1: False positive rate (FPR) for the compared workflows along ten replications for the three simulated scenarios. Panels correspond to: **A)** S1, **B)** S2 and **C)** S3 for DIE pipelines, and, **D)** S1**, E)** S2 and **F)** S3 for DS workflows.



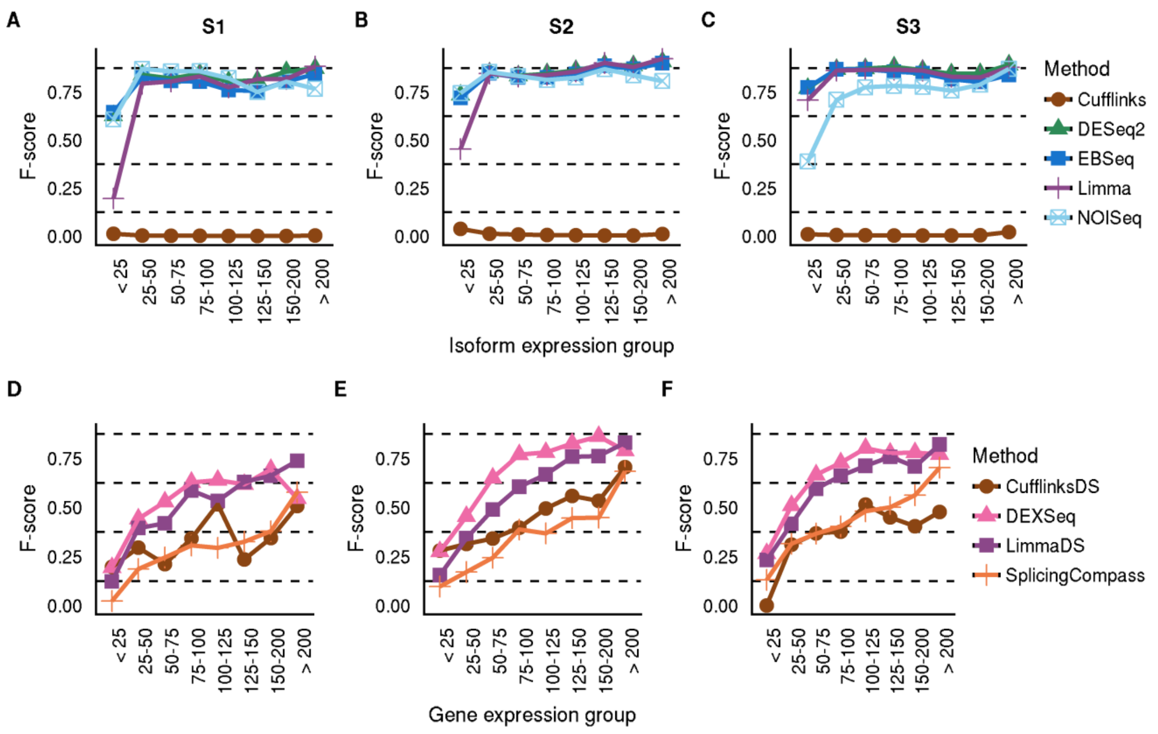
Supplementary Figure S2: Distribution of false positives (FP) along gene groups defined according to the number of isoforms per gene. Panels **A, B**, and **C** show the results for the DIE workflows and panels **D, E**, and **F** for DS workflows, in S1, S2 and S3 correspondingly.



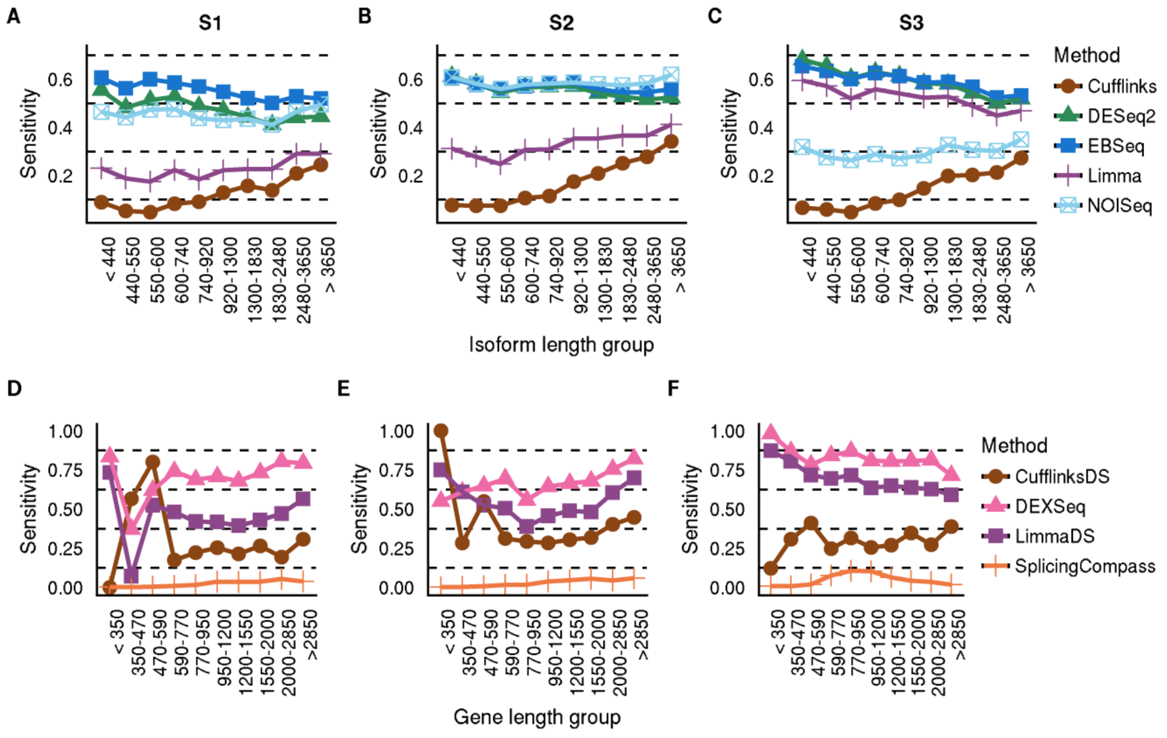
Supplementary Figure S3: Average sensitivity evaluated on groups of isoforms/genes according to their expression level. Gene groups are distributed along the x-axis**.** Panels **A** to **C** correspond to the results of DIE pipelines on S1, S2 and S3 scenarios. Panels **D** to **F** illustrate DS workflows results.



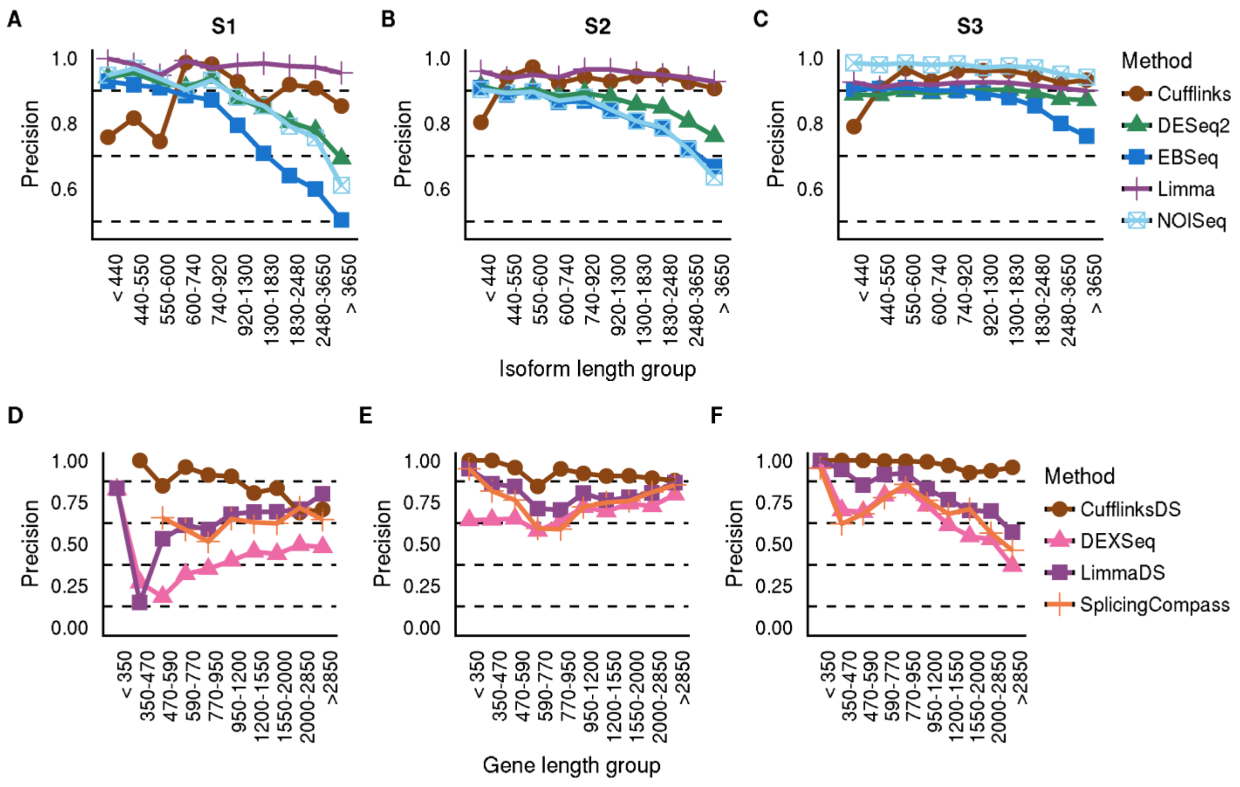
Supplementary Figure S4: Average precision evaluated on groups of isoforms/genes according to their expression level. Gene groups are distributed along the x-axis**.** Panels **A** to **C** correspond to the results of DIE pipelines on S1, S2 and S3 scenarios. Panels **D** to **F** illustrate DS workflows results.



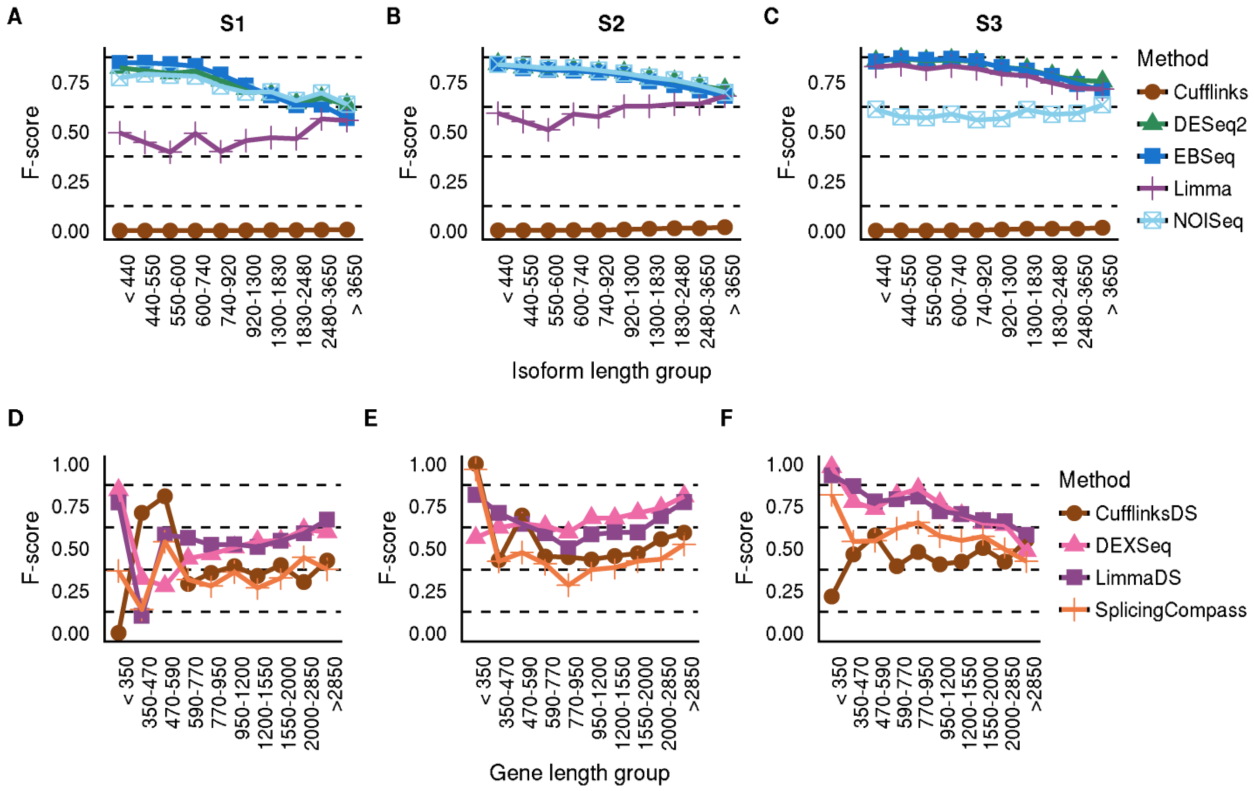
Supplementary Figure S5: Average F-score evaluated on groups of isoforms/genes according to their expression level. Gene groups are distributed along the x-axis**.** Panels **A** to **C** correspond to the results of DIE pipelines on S1, S2 and S3 scenarios. Panels **D** to **F** illustrate DS workflows results.



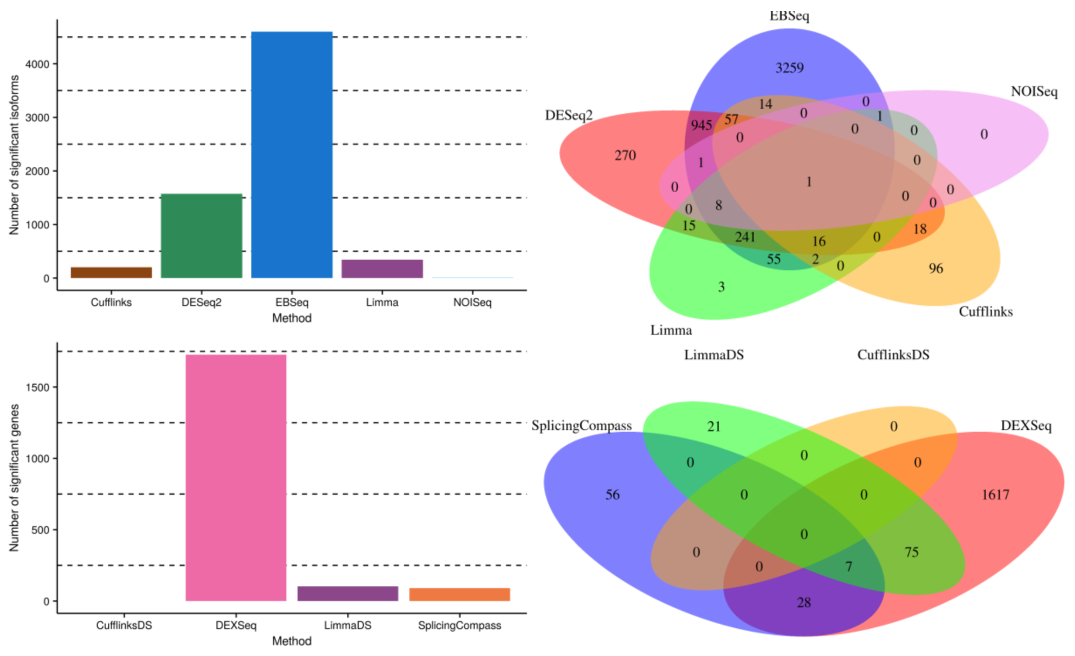
Supplementary Figure S6: Average sensitivity evaluated on groups of isoforms/genes according to their length. Gene groups are distributed along the x-axis**.** Panels **A** to **C** correspond to the results of DIE pipelines on S1, S2 and S3 scenarios. Panels **D** to **F** illustrate DS workflows results.



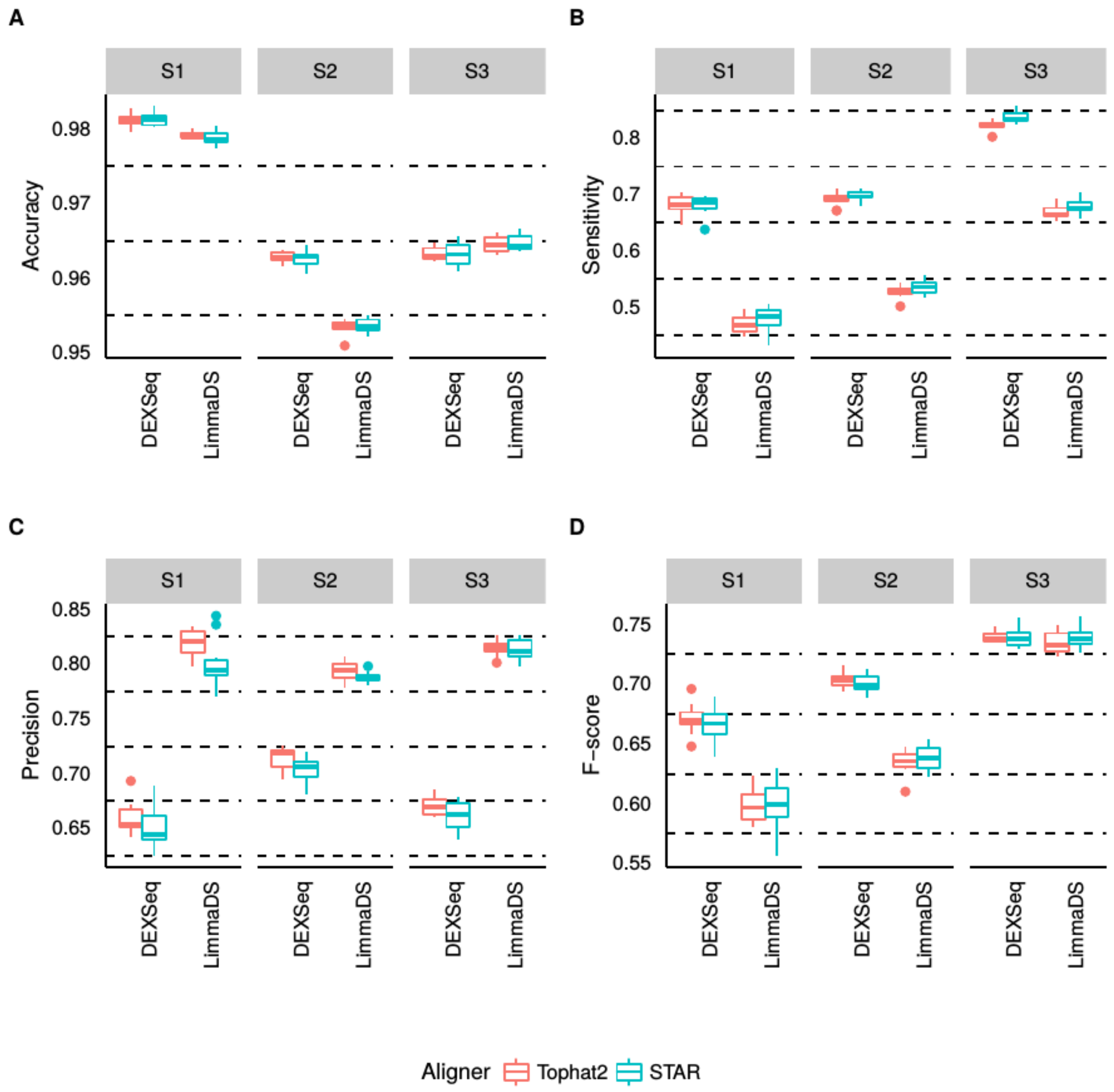
Supplementary Figure S7: Average precision evaluated on groups of isoforms/genes according to their length. Gene groups are distributed along the x-axis**.** Panels **A** to **C** correspond to the results of DIE pipelines on S1, S2 and S3 scenarios. Panels **D** to **F** illustrate DS workflows results.



Supplementary Figure S8: Average F-score evaluated on groups of isoforms/genes according to their length. Gene groups are distributed along the x-axis**.** Panels **A** to **C** correspond to the results of DIE pipelines on S1, S2 and S3 scenarios. Panels **D** to **F** illustrate DS workflows results.



Supplementary Figure S9: Number of differentially expressed isoforms and spliced genes detected on real samples. Bar plots indicates the number of significant isoforms (top) /genes (bottom) detected by means of the DIE and DS workflows, respectively. The Venn diagrams illustrate the amount of significant isoforms/genes shared between pipelines. Top graphics correspond to DIE workflows whereas bottom plots represented DS pipelines results.



APPENDIX I

Analysis Steps:

1- Samples downloading: Download all samples from the NCBI site https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE22260. The Normal samples were: SRR057649, SRR057650, SRR057651, SRR057652, SRR057653, SRR057654, SRR057655, SRR057656, SRR057657 and SRR057658; whereas, the Tumor samples were: SRR057631, SRR057633, SRR057639, SRR057640, SRR057643, SRR057644, SRR057645, SRR057646, SRR057647 and SRR057648.

2- Transcriptome alignment: Each sample was aligned against the HG19 reference trasncriptome using `Bowtie`. Then, aligned reads achieving maping q-value lower than 20 were filtered out. Command lines:

```
> cd path_to_sample_alignment_cdna

> bowtie -q -n 3 -e 99999999 -l 15 -I 30 -X 270 --chunkmbs 512 -p 12 -a -S
/path_to_transcriptome_reference/transcriptome_reference -1
/path_fastq_reads/sample_1.fastq -2 /path_fastq_reads/sample_2.fastq
sample.sam

> samtools view -bh -q20 -S sample.sam > sample.mapq20.bam
```

3- Quantification at isoform level: Aligned reads were quantified to obtain the expression profile of each sample at the isoform level. Command line:

```
> cd path_to_RSEM_real_sample

> rsem-calculate-expression --paired-end --no-bam-output -p 18 --samtools-
sort-mem 8G --temporary-folder tmp --bam
/path_to_sample_alignment_cdna/sample.mapq20.bam
/path_to_transcriptome_reference/transcriptome_reference sample > out.txt
```

4- Modification of expression profiles: The sample's expression profiles were used to build the expression matrix where differential expression changes were simulated. This step was performed running the R script: "profilesSimulation.R" listed in the **APPENDIX II**.

5- Samples simulation: Each sample for every simulation of each scenario was generated using the next command line, whereas N means the number of simulated reads:

```
> cd path_to_simulated_sample_sim_scenario

> rsem-simulate-reads
/path_to_transcriptome_reference/transcriptome_reference
/path_to_RSEM_real_sample/sample.stat/sample.model
/path_to_simulated_profile_sample/sample_sim_iso.results 0.1 N sample
```

6- Bowtie alignment of simulated samples: Each simulated sample was aligned against the HG19 reference trasncriptome. Then, the reads achieving maping q-value lower than 20 were filtered out. Command lines:

```
> cd path_to_alignment_cdna_sim_scenario
```

```
> bowtie -q -n 3 -e 99999999 -l 15 -I 30 -X 270 --chunkmbs 512 -p 12 -a -S
/path_to_transcriptome_reference/transcriptome_reference -1
/path_to_simulated_sample_sim_scenario/sample_1.fq -2
/path_to_simulated_sample_sim_scenario /sample_2.fq sample.sam
```

```
> samtools view -bh -q20 -S sample.sam > sample.mapq20.bam
```

7- Quantification of simulated samples at isoform level: Aligned reads were quantified to obtain the expression profile of each sample at the isoform level. Command line:

```
> cd path_to_RSEM_sim_scenario
```

```
> rsem-calculate-expression --paired-end --no-bam-output -p 18 --samtools-
sort-mem 8G --temporary-folder tmp --bam
/path_to_alignment_cdna_sim_scenario/sample.mapq20.bam
/path_to_transcriptome_reference/transcriptome_reference sample > out.txt
```

8- Tophat alignment of simulated samples: Each simulated sample was aligned against the HG19 reference genome. Then, the reads achieving maping q-value lower than 20 were filtered out. Command lines:

```
> cd path_to_alignment_genome_sim_scenario
```

```
> tophat2 --bowtie1 --no-novel-juncs --no-novel-indels --segment-length 18 -r
800 -p 22 --no-coverage-search -o sample --transcriptome-index
/path_to_transcriptome_reference/transcriptome_reference
/path_to_genome_reference/genome_reference
/path_to_simulated_sample_sim_scenario/sample_1.fq
/path_to_simulated_sample_sim_scenario/sample_2.fq
```

9- Exon level quantification for SplicingCompass: This quantification was made using using the coverageBed tool. It was based on a flattened reference annotation file generated by means of an R script provided with the SplicingCompass package.

```
> cd path_to_quantification_covBed_sim_scenario
```

```
> coverageBed -split -abam
/path_to_alignment_genome_sim_scenario/sample/accepted_hits.bam -b
/path_to_annotation_files/flattened.splcmp.gtf > sample.covBed.counts
```

10- Exon level quantification for DEXSeq: This quantification was made using using the python script `dexseq_count.py` provided with the DEXSeq package. It was based on a flattened reference annotation file generated by means of the other python script `dexseq_prepare_annotation.py`, also provided with the DEXSeq package.

```
> cd path_to_quantification_DEXSeq_sim_scenario
```

```
> python /usr/local/lib/R/library/DEXSeq/python_scripts/dexseq_count.py -p
yes -f bam -a 20 -r pos /path_to_annotation_files/flattened.dexseq.gtf
/path_to_alignment_genome_sim_scenario/sample/accepted_hits.bam
sample.htseq.counts
```

11- Analysis with Cufflinks: This steps involved several stages. I a first stage, the quantification was done using Cufflinks; then, samples assemblies were merged with Cuffmerge and compared against the annotation file using Cuffcompare. Finally, differential expression changes were analyzed by means of Cuffdiff.

```
> cd cufflinks_sample_sim_scenario
```

```
> cufflinks -p16 -g /path_to_annotation_files/annotation.gtf -b
/path_to_genome_reference/genome_reference -L sample
/path_to_alignment_genome_sim_scenario/sample/accepted_hits.bam
```

```
> cd cufflinks_sim_scenario
```

```
> cuffmerge -p 18 -s /path_to_genome_reference/genome_reference -g
/path_to_annotation_files/annotation.gtf assemblies.txt
```

```
> cuffcompare -p 18 -r /path_to_annotation_files/annotation.gtf -s
/path_to_genome_reference/genome_reference merged_asm/merged.gtf
/path_to_annotation_files/annotation.
```

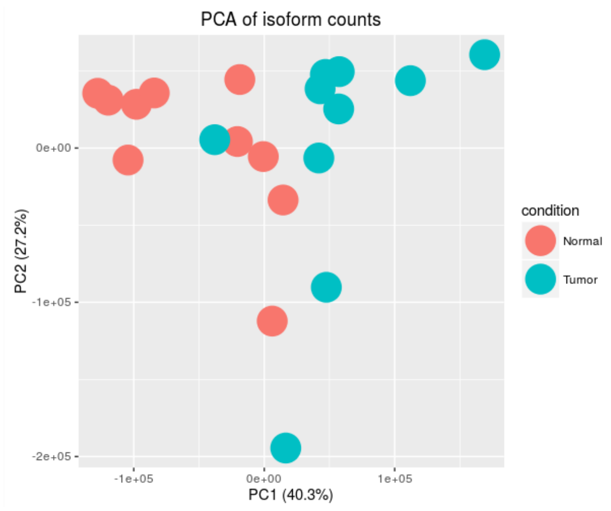
```
> cuffdiff -p 20 -o cuffdiff_sim_scenario -L C,T -u cuffcmp.combined.gtf
/path_to_alignment_genome_sim_scenario/sample1C/accepted_hits.bam,
/path_to_alignment_genome_sim_scenario/sample2C/accepted_hits.bam,
/path_to_alignment_genome_sim_scenario/sample3C/accepted_hits.bam,
/path_to_alignment_genome_sim_scenario/sample4C/accepted_hits.bam
/path_to_alignment_genome_sim_scenario/sample1T/accepted_hits.bam,
/path_to_alignment_genome_sim_scenario/sample2T/accepted_hits.bam,
/path_to_alignment_genome_sim_scenario/sample3T/accepted_hits.bam,
/path_to_alignment_genome_sim_scenario/sample4T/accepted_hits.bam,
```

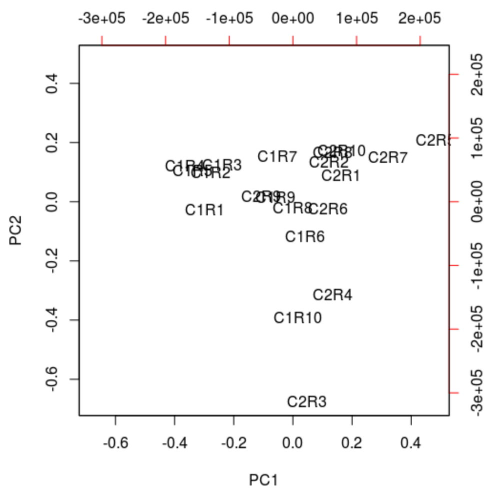
12- Differential expression analysis: This was the last step. It was performed running the DIEAnalysis.R and DSAnalysis.R scripts listed in the **APPENDIX III** and **APPENDIX IV**, respectively.

APPENDIX II

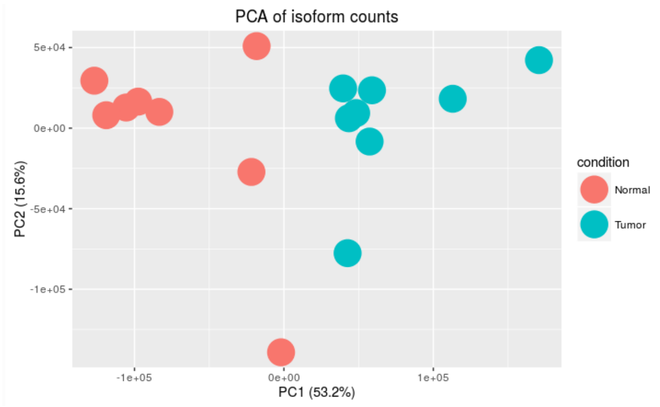
```
# profilesSimulation.R
# This script was designed to perform ten simulations of one experimental
scenario were X replicates per condition and Y% of deregulated genes were
considered. In S1, X=4 and Y=5; in S2, X=4 and Y=10; and in S3, X=8 and Y=10.
library(BiocParallel);library(ggplot2);lirbary(gridExtra)
setwd("/path_to_simulation_scenario")
# read file containing gene/isoforms definition
g2t<-read.delim("gene2transc.txt", header=FALSE)
head(g2t,3)
#             gene_id     transcript_id
# 92274 ENSG00000000003 ENST00000373020
# 92275 ENSG00000000003 ENST00000496771
# 92276 ENSG00000000003 ENST00000494424
## load RSEM quantification for all samples generated in step 3
#gene counts
gene_files<-paste("SRR0576",c(49:58,31,33,39,40,43:48), ".genes.results",
sep="")
gene_res<-lapply(gene_files, function(gene_f){
  return(read.delim(gene_f))
})
names(gene_res)<-sapply(gene_files,function(gene_f){
  return(strsplit(gene_f, split="[.]")[[1]][1])
})
gene_cm<-as.data.frame(do.call(cbind, lapply(1:length(gene_res), function(y){
  gene_sample<-gene_res[[y]]
  return(gene_sample$expected_count)
})))
rownames(gene_cm)<-gene_res[[1]]$gene_id
colnames(gene_cm)<-names(gene_res)
gene_tpm<-as.data.frame(do.call(cbind, lapply(1:length(gene_res), function(y){
  gene_sample<-gene_res[[y]]
  return(gene_sample$TPM )
})))
rownames(gene_tpm)<-gene_res[[1]]$gene_id
colnames(gene_tpm)<-names(gene_res)
dim(gene_cm)
# [1] 44464    20
# isoform counts
iso_files<-paste("SRR0576",c(49:58,31,33,39,40,43:48), ".isoforms.results",
sep="")
iso_res<-lapply(iso_files, function(iso_f){
  return(read.delim(iso_f))
})
names(iso_res)<-sapply(iso_files,function(iso_f){
  return(strsplit(iso_f, split="[.]")[[1]][1])
})
iso_cm<-as.data.frame(do.call(cbind, lapply(1:length(iso_res), function(x){
  iso_sample<-iso_res[[x]]
  return(iso_sample$expected_count)
})))
rownames(iso_cm)<-iso_res[[1]]$transcript_id
colnames(iso_cm)<-c(paste("C1R", c(1:10), sep=""), paste("C2R", c(1:10),
sep=""))
iso_tpm<-as.data.frame(do.call(cbind, lapply(1:length(iso_res), function(x){
  iso_sample<-iso_res[[x]]
  return(iso_sample$TPM)
})))
rownames(iso_tpm)<-iso_res[[1]]$transcript_id
colnames(iso_tpm)<-c(paste("C1R", c(1:10), sep=""), paste("C2R", c(1:10),
sep=""))
dim(iso_cm)
# [1] 169481     20
##Filter not-expressed genes
index_g<-unique(do.call(c,sapply(1:nrow(gene_tpm), function(x){
   if(any(gene_tpm[x,1:10] ==0) | any(gene_tpm[x,11:20] ==0) ) {
    return(x)
    }else return(NULL)
})))
length(index_g)
# [1] 28280
gene_cm<-gene_cm[-index_g,]
dim(gene_cm)
# [1] 16184    20
gene_tpm<-gene_tpm[-index_g,]
dim(gene_tpm)
# [1] 16184    20
g2t_ok<-g2t[(g2t$gene_id%in% rownames(gene_cm)),]
iso_cm<-iso_cm[rownames(iso_cm) %in% g2t_ok$transcript_id,]
dim(iso_cm)
# [1] 119965     20
iso_tpm<-iso_tpm[rownames(iso_tpm) %in% rownames(iso_cm),]
# Exploring conditions separability by means of a PCA plot.
pr_comp_y<-prcomp(t(iso_cm))
resumen <- summary(pr_comp_y)
labX <- signif((resumen$importance[2,1])*100, 3)
labY <- signif((resumen$importance[2,2])*100, 3)
```

```
PCAdata<-cbind(as.data.frame(pr_comp_y$x), condition=factor(c(rep("Normal",
10), rep("Tumor",10)), levels=c("Normal", "Tumor")))
ggplot(PCAdata, aes(x=PC1, y=PC2, fill=condition, color=condition)) +
labs(title="PCA of isoform counts", x=paste("PC1 (",labX,"%)", sep=""),
y=paste("PC2 (",labY,"%)", sep="")) + geom_point(aes(fill = condition),
size=10)
biplot(pr_comp_y,cex=c(1,0.001),xlabs=colnames(iso_tpm),ylabs=rep("",
nrow(iso_tpm)), var.axes=FALSE)
```

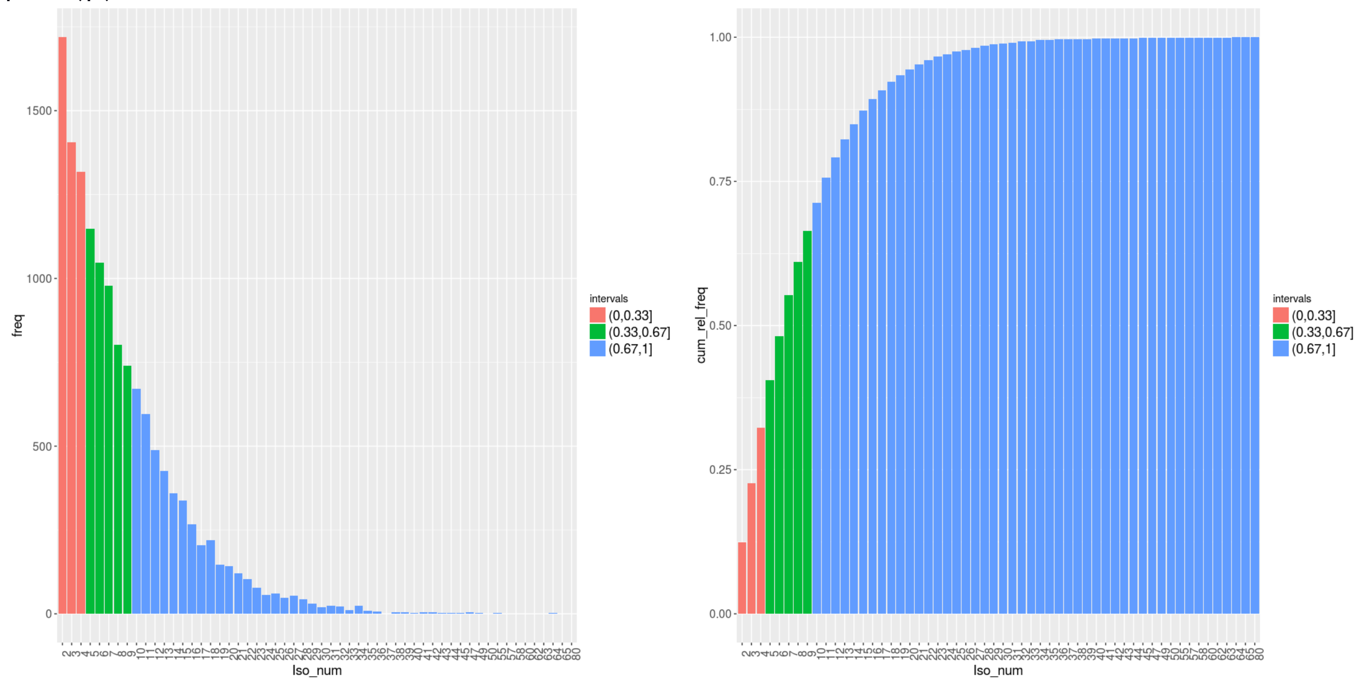




```
# The samples identified as outlier were C1R6, C1R10, C2R3 and C2R9, which
correspond to samples SRR057654, SRR057657, SRR057633 and SRR057647, respectively.
pr_comp_y<-prcomp(t(iso_cm[, c(1:5,7:9,11:12,14:18,20)]))
resumen <- summary(pr_comp_y)
labX <- signif((resumen$importance[2,1])*100, 3)
labY <- signif((resumen$importance[2,2])*100, 3)
PCAdata<-cbind(as.data.frame(pr_comp_y$x), condition=factor(c(rep("Normal",8),
rep("Tumor",8)), levels=c("Normal", "Tumor")))
ggplot(PCAdata, aes(x=PC1, y=PC2, fill=condition,
color=condition))+labs(title="PCA of isoform counts", x=paste("PC1
(",labX,"%)", sep=""), y=paste("PC2 (",labY,"%)", sep="")) +
geom_point(aes(fill = condition), size = 10)
```

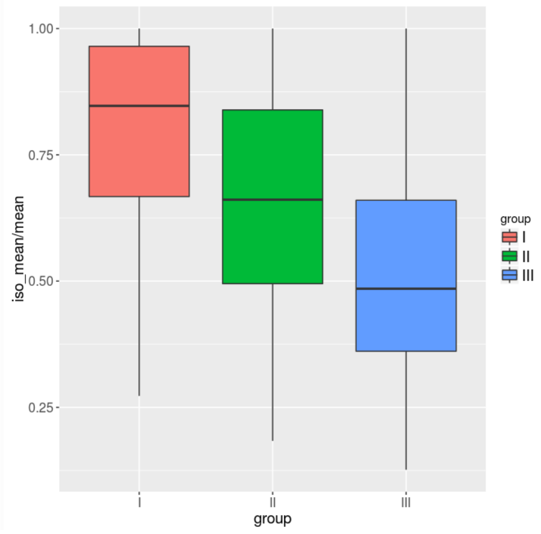


```
# For S1 and S2, we used: SRR057649, SRR057650, SRR057651 and SRR057652 for
condition C and SRR057631, SRR057643, SRR057645 and SRR057648 for T. For S3 we
used the 16 samples preserved after outlier samples removal
#S1-S2
iso_cm<-iso_cm[, c(1,2,3,4,11,15,17,20)]
iso_tpm<-iso_tpm[, c(1,2,3,4,11,15,17,20)]
gene_cm<-gene_cm[, c(1,2,3,4,11,15,17,20)]
gene_tpm<-gene_tpm[, c(1,2,3,4,11,15,17,20)]
colnames(iso_tpm)<- colnames(iso_cm)<-colnames(gene_tpm)<-colnames(gene_cm)<-
c(paste("C1R", 1:4, sep="") , paste("C2R", 1:4, sep="" ))
# S3
# iso_cm<-iso_cm[, c(1:5,7:9,11:12,14:18,20)]
# iso_tpm<-iso_tpm[,c(1:5,7:9,11:12,14:18,20)]
# gene_cm<-gene_cm[,c(1:5,7:9,11:12,14:18,20)]
# gene_tpm<-gene_tpm[,c(1:5,7:9,11:12,14:18,20)]
# colnames(iso_tpm)<- colnames(iso_cm)<-colnames(gene_tpm)<-colnames(gene_cm)<-
c(paste("C1R", 1:8, sep="") , paste("C2R", 1:8, sep="" ))
#Characterization of the number of isoforms per gene
genes<-as.character(unique(g2t_ok$gene_id))
genes<-sort(genes)
numb_iso<-do.call(c, bplapply(genes, function(gen){
  return(length(g2t[g2t$gene_id == gen,"transcript_id"]))
  }, BPPARAM=MulticoreParam(20)))
#construct a table with gene information
gene_info<-data.frame(gene_id=genes, numb_iso=numb_iso)
freq<-data.frame(table(gene_info$numb_iso[!gene_info$numb_iso ==1]),
accum=cumsum(table(gene_info$numb_iso[!gene_info$numb_iso==1] ))/sum(table(
gene_info$numb_iso[ !gene_info$numb_iso==1])))
colnames(freq)<-c("Iso_num", "freq", "cum_rel_freq")
freq[,"score"]<-cut(freq[,"cum_rel_freq"], breaks=c(0,0.33,0.67,1),
include.lowest=FALSE, right=TRUE)
g1<-ggplot(freq,aes(x=Iso_num, y=freq, fill=as.factor(score))) + geom_bar(
stat="identity")+guides(fill=guide_legend(title="intervals"))+
theme(axis.text=element_text(size=12), axis.title=element_text(size=14),
legend.text=element_text(size=14))
g2<-ggplot(freq,aes(x=Iso_num, y=cum_rel_freq,fill=as.factor(score))) +
geom_bar(stat="identity")+scale_fill_hue()+guides(fill=guide_legend(
title="intervals"))+theme(axis.text=element_text(size=12),
axis.title=element_text(size=14), legend.text=element_text(size=14))
p<-arrangeGrob(g1,g2, nrow=1, ncol=2)
plot(p)
```



```
# genes were clustered in four groups according to the number of annotated
transcripts #that they have:
# “0” = 1 annotated transcript
# “I” = 2-4 annotated transcripts
# “II”= 5-9 annotated transcripts
# “III”= >9 annotated transcripts
# add the gene groups to the gene_info data.frame
gene_info$group<-cut(gene_info[,"numb_iso"], breaks=c(1,2,5,10,Inf),
include.lowest=TRUE, right=FALSE)
levels(gene_info$group)<-c("0","I", "II", "III")
table(gene_info$group)
#    0    I   II  III
# 2397 4444 4717 4626
# add quantification along replicates
gene_info$meanC1<-rowMeans(gene_cm[,1:4])
gene_info$meanC2<-rowMeans(gene_cm[,5:8])
gene_info$mean<-rowMeans(gene_cm)
gene_info$varC1<-apply(gene_cm[,1:4],1,var)
```

```
gene_info$varC2<-apply(gene_cm[,5:8],1,var)
gene_info$var<-apply(gene_cm,1,var)
# create a table with isoform information
iso_info<-merge(x=g2t_ok ,y=gene_info, by="gene_id", sort=FALSE)
iso_info<-iso_info[order(iso_info$gene_id,
iso_info$transcript_id),c(2,1,3:ncol(iso_info))]
iso_info$iso_meanC1<-rowMeans(iso_cm[,1:4]);
iso_info$iso_meanC2<-rowMeans(iso_cm[,5:8])
iso_info$iso_mean<-rowMeans(iso_cm)
iso_info$iso_varC1<-apply(iso_cm[,1:4],1,var)
iso_info$iso_varC2<-apply(iso_cm[,5:8],1,var)
iso_info$iso_var<-apply(iso_cm,1,var)
iso_info$ratiosC1<-iso_info$iso_meanC1/iso_info$meanC1
iso_info$ratiosC2<-iso_info$iso_meanC2/iso_info$meanC2
# determining the major isoform for each gene
id_mayor<-do.call(c, bplapply(1:length(unique(g2t_ok$gene_id)), function(x){
  maxM<-which.max(iso_info$iso_mean[iso_info$gene_id == as.character(
unique(g2t_ok$gene_id)[x])])
  return(as.character(iso_info$transcript_id[iso_info$gene_id ==
as.character(unique(g2t_ok$gene_id)[x])])[maxM])
  }, BPPARAM=MulticoreParam(22)))
g2t_ok$mayor<-g2t_ok$transcript_id %in% id_mayor
# proportion of major isoforms
ggplot(iso_info[iso_info$transcript_id%in%id_mayor,],aes(x=group,
y=iso_mean/mean,fill=as.factor(group)))+geom_boxplot()+scale_fill_hue() +
guides(fill=guide_legend(title="group"))+labs(title="")+theme(
axis.text=element_text(size=12),axis.title=element_text(size=14),
legend.text=element_text(size=14))
```



```
##  Selecting target genes
# We only consider those genes having at least a mean tpm of 20 in condition 1!
```

```
index_exp<-unique(do.call(c,sapply(1:nrow(gene_tpm), function(x){
   if(mean(gene_tpm[x,1:4] >=20)){
    return(x)
    }else return(NULL)
})))
names_NSAgenes<-as.character(gene_info$gene_id[index_exp][gene_info[
index_exp,]$group=="0"])
names_SAgenesgI<-as.character(gene_info$gene_id[index_exp][gene_info[
index_exp,]$group=="I"])
names_SAgenesgII<-as.character(gene_info$gene_id[index_exp][gene_info[
index_exp,]$group=="II"])
names_SAgenesgIII<-as.character(gene_info$gene_id[index_exp][gene_info[
index_exp,]$group=="III"])
# We simulated 5% (S1) and 10% (S2 and S3) of total genes as deregulated (DE).
As total genes were 16184, we chosen 834 genes in the first case and 1560 in
the last case.
# We selected 120 (250) genes from 0 group and 208(480) from  I , II and III
groups.
DSI <- sample( 1:length(names_SAgenesgI),208)
DSgI<-names_SAgenesgI[DSI]
DSII <- sample( 1:length(names_SAgenesgII),208)
DSgII<-names_SAgenesgII[DSII]
DSIII <- sample( 1:length(names_SAgenesgIII),208)
DSgIII<-names_SAgenesgIII[DSIII]
DS_index<-data.frame(DSgI=DSgI, DSgII=DSgII, DSgIII=DSgIII)
DE_index<-  sample( 1:length(names_NSAgenes),120)
DEgenes<-names_NSAgenes[DE_index]
DSgenes<-c(DSgI, DSgII, DSgIII)
DS_iso<-iso_info[iso_info$gene_id %in% as.character(DSgenes),]
DE_iso<-iso_info[iso_info$gene_id %in% as.character(DEgenes),]
# Simulation groups: We selected 120 (250) genes from group “0” (1 isoform) to
be simulated as DE, and 208(480) from DSgenes to be simulated as DIE, DIEDS,
DS.
DIE_index<-sample(1:length(DSgenes), 208)
DIE_genes<-DSgenes[DIE_index]
table(gene_info$group[gene_info$gene_id %in% DIE_genes])
#   0   I  II III
#   0  72  67  69
SDg_index<-sample(1:length(DSgenes[-DIE_index]), 208)
SDg_genes<-DSgenes[-DIE_index][SDg_index]
table(gene_info$group[gene_info$gene_id %in% SDg_genes])
#   0   I  II III
#   0  72  67  69
SDDIE_genes<-DSgenes[-DIE_index][-SDg_index]
```

```
table(gene_info$group[gene_info$gene_id %in% SDDIE_genes])
#   0   I  II III
#   0  64  74  70
gene_info$DIE<-gene_info$gene_id %in% DIE_genes
gene_info$DS<-gene_info$gene_id %in% SDg_genes
gene_info$DIEDS<-gene_info$gene_id %in% SDDIE_genes
gene_info$DE<-gene_info$gene_id %in% DEgenes

##SIMULATION PROFILES
# 1) Differential isoforms expression
folds<-data.frame(a=2,b=3,d=4,e=5)
# 52 (120) genes for each fold, and 26 (60) to have up regulation and 26 (60)
down regulated
a<-sample(DIE_genes, 52)
b<-sample(DIE_genes[!DIE_genes %in% a],52)
d<-sample(DIE_genes[!DIE_genes %in% a & !DIE_genes %in% b],52)
e<-DIE_genes[!DIE_genes %in% a & !DIE_genes %in% b & !DIE_genes %in% d]
DIE_genesdf<-data.frame(a=a, b=b,d=d,e=e)
gene_info$DIEgroup<-factor(rep(0, nrow(gene_info)), levels=c(0, folds$a,
1/folds$a, folds$b, 1/folds$b,folds$d,1/folds$d,folds$e,1/folds$e))
gene_info$DIEgroup[gene_info$gene_id %in% DIE_genesdf[,1]]<-factor(rep(c(
folds$a, 1/folds$a), 26), levels=c(folds$a,1/folds$a))
gene_info$DIEgroup[gene_info$gene_id %in% DIE_genesdf[,2]]<-factor(rep(c(
folds$b,1/folds$b), 26), levels=c(folds$b,1/folds$b))
gene_info$DIEgroup[gene_info$gene_id %in% DIE_genesdf[,3]]<-factor(rep(c(
folds$d,1/folds$d), 26), levels=c(folds$d,1/folds$d))
gene_info$DIEgroup[gene_info$gene_id %in% DIE_genesdf[,4]]<-factor(rep(c(
folds$e,1/folds$e), 26), levels=c(folds$e,1/folds$e))

## 2) Differential splicing
ratios<-c("0.8-0.5","0.5-0.8","0.6-0.3","0.3-0.6","0.4-0.1","0.1-0.4","0.7-0",
"0-0.7")
#For the ratio group “0.1-0.4” (“0.4-0.1”) we considered only genes from groups
II and III, meanwhile for ratio group “0-0.7” (“0.7-0”) we considered only
genes from I group in order to simulate major isoforms proportions observed in
a real situation.
# 68 genes for the first and second ratio groups and 72 for the last two. In S2
and S3 160 genes for each ratio groups were used
a<-sample(SDg_genes, 68) #
b<-sample(SDg_genes[!SDg_genes %in% a],68)
d<-SDg_genes[!SDg_genes %in% a & !SDg_genes %in% b]
DS_genesdf<-data.frame(a="", b="",d=d)
levels(DS_genesdf$a)<-c("", a)
DS_genesdf$a[1:length(a)]<-a
```

```
levels(DS_genesdf$b)<-c("", b)
DS_genesdf$b[1:length(b)]<-b
gene_info$DSgroup<-0
gene_info$DSgroup[gene_info$gene_id %in% DS_genesdf$a]<-c(rep(ratios[1],34),
rep(ratios[2], 34))
gene_info$DSgroup[gene_info$gene_id %in% DS_genesdf$b]<-(c(rep(ratios[3],34),
rep(ratios[4], 34)))
gene_info$DSgroup[gene_info$gene_id %in% DS_genesdf$d&gene_info$group!= "I"]<-
(c(rep(ratios[5],21), rep(ratios[6], 21)))
gene_info$DSgroup[gene_info$gene_id %in% DS_genesdf$d&gene_info$group=="I" ]<-
(c(rep(ratios[7],15), rep(ratios[8], 15)))

## 3) Differential isoforms expression ad differential splicing
folds<-c("2-0.8-0.5","0.5-0.8-0.5", "2-0.8-0.3", "4-0.8-0.5", "4-0.8-0.3")
# 41 genes for each fold and 44 for the 0.5-0.8-0.5. In S2 and S3, 96 for each
fold were used
a<-sample(SDDIE_genes, 41)
b<-sample(SDDIE_genes[!SDDIE_genes %in% a],44)
d<-sample(SDDIE_genes[!SDDIE_genes %in% a & !SDDIE_genes %in% b],41)
e<-sample(SDDIE_genes[!SDDIE_genes %in% a & !SDDIE_genes %in% b & !SDDIE_genes
%in% d],41)
f<-SDDIE_genes[!SDDIE_genes %in% a & !SDDIE_genes %in% b & !SDDIE_genes %in% d
& !SDDIE_genes %in% e]
SDDIE_genesdf<-data.frame(a="", b=b,d="",e="", f="")
levels(SDDIE_genesdf$a)<-c("", a)
SDDIE_genesdf$a[1:length(a)]<-a
levels(SDDIE_genesdf$d)<-c("", d)
SDDIE_genesdf$d[1:length(d)]<-d
levels(SDDIE_genesdf$e)<-c("", e)
SDDIE_genesdf$e[1:length(e)]<-e
levels(SDDIE_genesdf$f)<-c("", f)
SDDIE_genesdf$f[1:length(f)]<-f
gene_info$DIEDSgroup<-factor(rep(0, nrow(gene_info)), levels=c(0,folds))
gene_info$DIEDSgroup[gene_info$gene_id %in% SDDIE_genesdf$a]<-folds[1]
gene_info$DIEDSgroup[gene_info$gene_id %in% SDDIE_genesdf$b]<-folds[2]
gene_info$DIEDSgroup[gene_info$gene_id %in% SDDIE_genesdf$d]<-folds[3]
gene_info$DIEDSgroup[gene_info$gene_id %in% SDDIE_genesdf$e]<-folds[4]
gene_info$DIEDSgroup[gene_info$gene_id %in% SDDIE_genesdf$f]<-folds[5]

## 4) DE genes
folds<-data.frame(a=2,b=4)
# 60 (125) genes for each fold, and 28 to have up regulation and 28 dow reg
a<-sample(DEgenes, 60)
b<-DEgenes[!DEgenes %in% a]
```

```
DEgenesdf<-data.frame(a=a, b=b)
gene_info$DEgroup<-factor(rep(0, nrow(gene_info)), levels=c(0, folds$a,
1/folds$a, folds$b, 1/folds$b))
gene_info$DEgroup[gene_info$gene_id %in% DEgenesdf$a]<-factor(c(rep(folds$a,
30), rep(1/folds$a, 30)), levels=c(folds$a,1/folds$a))
gene_info$DEgroup[gene_info$gene_id %in% DEgenesdf$b]<-factor(c(rep(folds$b,
30), rep(1/folds$b, 30)), levels=c(folds$b,1/folds$b))
iso_info<-merge(x=gene_info[,c(1,10:17)], y=iso_info, by="gene_id")
iso_info<-iso_info[,c(10,1,11:ncol(iso_info), 2:9)]

## Modifying distributional parameters values
# take as reference expression values from condition C (C1)
iso_info$meanC1Sim<-iso_info$iso_meanC1
iso_info$meanC2Sim<-iso_info$iso_meanC1
iso_info$varC1Sim<-iso_info$iso_varC1
iso_info$varC2Sim<-iso_info$iso_varC1
iso_info$mayor<-(iso_info$transcript_id %in% g2t_ok$transcript_id[ g2t_ok$mayor
])

## 1) Differential isoform expression
fold<-2:5
invFold<- c(0.5,0.333333333333333,0.25,0.2)
iso_info$meanC2Sim[iso_info$DIE & iso_info$DIEgroup %in% fold]<-
iso_info$meanC2Sim[iso_info$DIE & iso_info$DIEgroup %in% fold] * as.numeric(
as.character(iso_info$DIEgroup[iso_info$DIE& iso_info$DIEgroup %in% fold]))
iso_info$meanC1Sim[iso_info$DIE & iso_info$DIEgroup %in% invFold] <-
iso_info$iso_meanC1[iso_info$DIE & iso_info$DIEgroup %in% invFold]/as.numeric(
as.character(iso_info$DIEgroup[ iso_info$DIE & iso_info$DIEgroup %in%
invFold]))
iso_info$varC2Sim[iso_info$DIE & iso_info$DIEgroup %in% fold]<-
iso_info$varC2Sim[iso_info$DIE & iso_info$DIEgroup %in% fold]*(as.numeric(
as.character(iso_info$DIEgroup[iso_info$DIE & iso_info$DIEgroup %in% fold])))^2
iso_info$varC1Sim[iso_info$DIE & iso_info$DIEgroup %in% invFold]<-
iso_info$iso_varC1[iso_info$DIE & iso_info$DIEgroup %in% invFold]*(1/
as.numeric(as.character(iso_info$DIEgroup[iso_info$DIE & iso_info$DIEgroup %in%
invFold])))^2

## 2) Differential splicing
#0.8-0.5 mayor isoform ratio 0.8 and ratio 0.5. the rest should have ratio (1-
0.8)/(n-1) and (1-0.5)/(n-1)
altGene<-as.character(iso_info$gene_id[iso_info$DSgroup == "0.8-0.5" &
iso_info$mayor])
#expression of major isoforms
```

```
iso_info$meanC1Sim[iso_info$DSgroup == "0.8-0.5" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0.8-0.5" & iso_info$mayor]*0.8
iso_info$varC1Sim[iso_info$DSgroup == "0.8-0.5" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0.8-0.5" & iso_info$mayor]*0.8^2
iso_info$meanC2Sim[iso_info$DSgroup == "0.8-0.5" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0.8-0.5" & iso_info$mayor]*0.5
iso_info$varC2Sim[iso_info$DSgroup == "0.8-0.5" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0.8-0.5" & iso_info$mayor]*0.5^2
# the expression values to the rest
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DSgroup == "0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DSgroup == "0.8-0.5" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.8)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DSgroup == "0.8-0.5" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.8)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DSgroup == "0.8-0.5" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.5)/(numb_iso-1)
  varC2_sim<-iso_info$varC1[iso_info$DSgroup == "0.8-0.5" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.5)/(numb_iso-1))^2
  index<-which(iso_info$DSgroup == "0.8-0.5" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo $index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]

#0.5-0.8

altGene <-as.character(iso_info$gene_id[iso_info$DSgroup == "0.5-0.8" &
iso_info$mayor]  )
iso_info$meanC1Sim[iso_info$DSgroup == "0.5-0.8" & iso_info$mayor]  <-
iso_info$meanC1[iso_info$DSgroup == "0.5-0.8" & iso_info$mayor]  *0.5
iso_info$varC1Sim[iso_info$DSgroup == "0.5-0.8" & iso_info$mayor]  <-
iso_info$varC1[iso_info$DSgroup == "0.5-0.8" & iso_info$mayor]  *0.5^2
iso_info$meanC2Sim[iso_info$DSgroup == "0.5-0.8" & iso_info$mayor]  <-
iso_info$meanC1[iso_info$DSgroup == "0.5-0.8" & iso_info$mayor]  *0.8
iso_info$varC2Sim[iso_info$DSgroup == "0.5-0.8" & iso_info$mayor]  <-
iso_info$varC1[iso_info$DSgroup == "0.5-0.8" & iso_info$mayor]  *0.8^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DSgroup == "0.5-0.8" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DSgroup == "0.5-0.8" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.5)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DSgroup == "0.5-0.8" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.5)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DSgroup == "0.5-0.8" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.8)/(numb_iso-1)
  varC2_sim<-iso_info$varC1[iso_info$DSgroup == "0.5-0.8" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.8)/(numb_iso-1))^2
  index<-which(iso_info$DSgroup == "0.5-0.8" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]

# group II: mayor isoform ratio 0.6 and ratio 0.3. the rest should have ratio
(1-0.6)/(n-1) and (1-0.3)/(n-1)
#0.6-0.3
altGene <-as.character(iso_info$gene_id[iso_info$DSgroup == "0.6-0.3" &
iso_info$mayor])
iso_info$meanC1Sim[iso_info$DSgroup == "0.6-0.3" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0.6-0.3" & iso_info$mayor]*0.6
iso_info$varC1Sim[iso_info$DSgroup == "0.6-0.3" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0.6-0.3" & iso_info$mayor]*0.6^2
iso_info$meanC2Sim[iso_info$DSgroup == "0.6-0.3" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0.6-0.3" & iso_info$mayor]*0.3
iso_info$varC2Sim[iso_info$DSgroup == "0.6-0.3" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0.6-0.3" & iso_info$mayor]*0.3^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DSgroup == "0.6-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DSgroup == "0.6-0.3" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.6)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DSgroup == "0.6-0.3" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.6)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DSgroup == "0.6-0.3" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.3)/(numb_iso-1)
  varC2_sim<-iso_info$varC1[iso_info$DSgroup == "0.6-0.3" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.3)/(numb_iso-1))^2
  index<-which(iso_info$DSgroup == "0.6-0.3" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
```

```
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo [,-1]

altGene <-as.character(iso_info$gene_id[iso_info$DSgroup == "0.3-0.6" &
iso_info$mayor]  )
iso_info$meanC1Sim[iso_info$DSgroup == "0.3-0.6" & iso_info$mayor]  <-
iso_info$meanC1[iso_info$DSgroup == "0.3-0.6" & iso_info$mayor]  *0.3
iso_info$varC1Sim[iso_info$DSgroup == "0.3-0.6" & iso_info$mayor]  <-
iso_info$varC1[iso_info$DSgroup == "0.3-0.6" & iso_info$mayor]  *0.3^2
iso_info$meanC2Sim[iso_info$DSgroup == "0.3-0.6" & iso_info$mayor]  <-
iso_info$meanC1[iso_info$DSgroup == "0.3-0.6" & iso_info$mayor]  *0.6
iso_info$varC2Sim[iso_info$DSgroup == "0.3-0.6" & iso_info$mayor]  <-
iso_info$varC1[iso_info$DSgroup == "0.3-0.6" & iso_info$mayor]  *0.6^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DSgroup == "0.3-0.6" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DSgroup == "0.3-0.6" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.3)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DSgroup == "0.3-0.6" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.3)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DSgroup == "0.3-0.6" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.6)/(numb_iso-1)
  varC2_sim<-iso_info$varC1[iso_info$DSgroup == "0.3-0.6" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.6)/(numb_iso-1))^2
  index<-which(iso_info$DSgroup == "0.3-0.6" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]

# group III: mayor isoform ratio 0.4 and ratio 0.1. the rest should have ratio
(1-0.4)/(n-1) and (1-0.1)/(n-1)
#0.4-0.1
altGene<-as.character(iso_info$gene_id[iso_info$DSgroup == "0.4-0.1" &
iso_info$mayor])

iso_info$meanC1Sim[iso_info$DSgroup == "0.4-0.1" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0.4-0.1" & iso_info$mayor]*0.4
iso_info$varC1Sim[iso_info$DSgroup == "0.4-0.1" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0.4-0.1" & iso_info$mayor]*0.4^2
```

```
iso_info$meanC2Sim[iso_info$DSgroup == "0.4-0.1" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0.4-0.1" & iso_info$mayor]*0.1
iso_info$varC2Sim[iso_info$DSgroup == "0.4-0.1" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0.4-0.1" & iso_info$mayor]*0.1^2
# now should set the expression values to the rest!!
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DSgroup == "0.4-0.1" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DSgroup == "0.4-0.1" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.4)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DSgroup == "0.4-0.1" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.4)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DSgroup == "0.4-0.1" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.1)/(numb_iso-1)
  varC2_sim<-iso_info$varC1[iso_info$DSgroup == "0.4-0.1" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.1)/(numb_iso-1))^2
  index<-which(iso_info$DSgroup == "0.4-0.1" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]
# 0.4-0.1
altGene<-as.character(iso_info$gene_id[iso_info$DSgroup == "0.1-0.4" &
iso_info$mayor])
iso_info$meanC1Sim[iso_info$DSgroup == "0.1-0.4" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0.1-0.4" & iso_info$mayor]*0.1
iso_info$varC1Sim[iso_info$DSgroup == "0.1-0.4" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0.1-0.4" & iso_info$mayor]*0.1^2
iso_info$meanC2Sim[iso_info$DSgroup == "0.1-0.4" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0.1-0.4" & iso_info$mayor]*0.4
iso_info$varC2Sim[iso_info$DSgroup == "0.1-0.4" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0.1-0.4" & iso_info$mayor]*0.4^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DSgroup == "0.1-0.4" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DSgroup == "0.1-0.4" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.1)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DSgroup == "0.1-0.4" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.1)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DSgroup == "0.1-0.4" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.4)/(numb_iso-1)
  varC2_sim<-iso_info$varC1[iso_info$DSgroup == "0.1-0.4" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.4)/(numb_iso-1))^2
  index<-which(iso_info$DSgroup == "0.1-0.4" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]

# group IV: mayor isoform ratio 0.7 and ratio 0. the rest should have ratio (1-
0.7)/(n-1) and (1)/(n-1)
#0.7-0
altGene<-as.character(iso_info$gene_id[iso_info$DSgroup == "0.7-0" &
iso_info$mayor])
iso_info$meanC1Sim[iso_info$DSgroup == "0.7-0" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0.7-0" & iso_info$mayor]*0.7
iso_info$varC1Sim[iso_info$DSgroup == "0.7-0" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0.7-0" & iso_info$mayor]*0.7^2
iso_info$meanC2Sim[iso_info$DSgroup == "0.7-0" & iso_info$mayor]<-0
iso_info$varC2Sim[iso_info$DSgroup == "0.7-0" & iso_info$mayor]<-0
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DSgroup == "0.7-0" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DSgroup == "0.7-0" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1-0.7)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DSgroup == "0.7-0" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1-0.7)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DSgroup == "0.7-0" & !iso_info$mayor &
  iso_info$gene_id == gen]*(1)/(numb_iso-1)
  varC2_sim<-iso_info$varC1[iso_info$DSgroup == "0.7-0" & !iso_info$mayor &
  iso_info$gene_id == gen]*((1)/(numb_iso-1))^2
  index<-which(iso_info$DSgroup == "0.7-0" & !iso_info$mayor & iso_info$gene_id
  == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]
# 0.7-0
altGene<-as.character(iso_info$gene_id[iso_info$DSgroup == "0-0.7" &
iso_info$mayor])
iso_info$meanC1Sim[iso_info$DSgroup == "0-0.7" & iso_info$mayor]<-0
iso_info$varC1Sim[iso_info$DSgroup == "0-0.7" & iso_info$mayor]<-0
```

```
iso_info$meanC2Sim[iso_info$DSgroup == "0-0.7" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DSgroup == "0-0.7" & iso_info$mayor]*0.7
iso_info$varC2Sim[iso_info$DSgroup == "0-0.7" & iso_info$mayor]<-
iso_info$varC1[iso_info$DSgroup == "0-0.7" & iso_info$mayor]*0.7^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
numb_iso<-unique(iso_info$numb_iso[iso_info$DSgroup == "0-0.7" &
!iso_info$mayor & iso_info$gene_id == gen])
meanC1_sim<-iso_info$meanC1[iso_info$DSgroup == "0-0.7" & !iso_info$mayor &
iso_info$gene_id == gen]*(1)/(numb_iso-1)
varC1_sim<-iso_info$varC1[iso_info$DSgroup == "0-0.7" & !iso_info$mayor &
iso_info$gene_id == gen]*((1)/(numb_iso-1))^2
meanC2_sim<-iso_info$meanC1[iso_info$DSgroup == "0-0.7" & !iso_info$mayor &
iso_info$gene_id == gen]*(1-0.7)/(numb_iso-1)
varC2_sim<-iso_info$varC1[iso_info$DSgroup == "0-0.7" & !iso_info$mayor &
iso_info$gene_id == gen]*((1-0.7)/(numb_iso-1))^2
index<-which(iso_info$DSgroup == "0-0.7" & !iso_info$mayor & iso_info$gene_id
== gen)
return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]

## 3) Differential isoform expression and differential splicing
altGene<-as.character(iso_info$gene_id[iso_info$DIEDSgroup == "2-0.8-0.5" &
iso_info$mayor])
iso_info$meanC1Sim[iso_info$DIEDSgroup == "2-0.8-0.5" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "2-0.8-0.5" & iso_info$mayor]*0.8
iso_info$varC1Sim[iso_info$DIEDSgroup == "2-0.8-0.5" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "2-0.8-0.5" & iso_info$mayor]*0.8^2
iso_info$meanC2Sim[iso_info$DIEDSgroup == "2-0.8-0.5" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "2-0.8-0.5" & iso_info$mayor]*(2*0.5)
iso_info$varC2Sim[iso_info$DIEDSgroup == "2-0.8-0.5" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "2-0.8-0.5" & iso_info$mayor]*(2*0.5)^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DIEDSgroup == "2-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "2-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*(1-0.8)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DIEDSgroup == "2-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.8)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "2-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.5)/(numb_iso-1)*2)
  varC2_sim<-iso_info$varC1[iso_info$DIEDSgroup == "2-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.5)/(numb_iso-1)*2)^2
  index<-which(iso_info$DIEDSgroup == "2-0.8-0.5" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[altGene $index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
altGene[,-1]

# 0.5-0.8-0.5
altGene<-as.character(iso_info$gene_id[iso_info$DIEDSgroup == "0.5-0.8-0.5" &
iso_info$mayor])
iso_info$meanC1Sim[iso_info$DIEDSgroup == "0.5-0.8-0.5" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "0.5-0.8-0.5" & iso_info$mayor]*2*0.8
iso_info$varC1Sim[iso_info$DIEDSgroup == "0.5-0.8-0.5" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "0.5-0.8-0.5" & iso_info$mayor]*(2*0.8)^2
iso_info$meanC2Sim[iso_info$DIEDSgroup == "0.5-0.8-0.5" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "0.5-0.8-0.5" & iso_info$mayor]*(0.5)
iso_info$varC2Sim[iso_info$DIEDSgroup == "0.5-0.8-0.5" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "0.5-0.8-0.5" & iso_info$mayor]*(0.5)^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DIEDSgroup == "0.5-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "0.5-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*(1-0.8)/(numb_iso-1)*2
  varC1_sim<-iso_info$varC1[iso_info$DIEDSgroup == "0.5-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.8)/(numb_iso-1)*2)^2
  meanC2_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "0.5-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.5)/(numb_iso-1))
  varC2_sim<-iso_info$varC1[iso_info$DIEDSgroup == "0.5-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.5)/(numb_iso-1))^2
  index<-which(iso_info$DIEDSgroup == "0.5-0.8-0.5" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]

# 2-0.8-0.3
altGene<-as.character(iso_info$gene_id[iso_info$DIEDSgroup == "2-0.8-0.3" &
iso_info$mayor])
```

```
iso_info$meanC1Sim[iso_info$DIEDSgroup == "2-0.8-0.3" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "2-0.8-0.3" & iso_info$mayor]*0.8
iso_info$varC1Sim[iso_info$DIEDSgroup == "2-0.8-0.3" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "2-0.8-0.3" & iso_info$mayor]*0.8^2
iso_info$meanC2Sim[iso_info$DIEDSgroup == "2-0.8-0.3" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "2-0.8-0.3" & iso_info$mayor]*(2*0.3)
iso_info$varC2Sim[iso_info$DIEDSgroup == "2-0.8-0.3" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "2-0.8-0.3" & iso_info$mayor]*(2*0.3)^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DIEDSgroup == "2-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "2-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen]*(1-0.8)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DIEDSgroup == "2-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.8)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "2-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.3)/(numb_iso-1)*2)
  varC2_sim<-iso_info$varC1[iso_info$DIEDSgroup == "2-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.3)/(numb_iso-1)*2)^2
  index<-which(iso_info$DIEDSgroup == "2-0.8-0.3" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]

# 4-0.8-0.5
altGene<-as.character(iso_info$gene_id[iso_info$DIEDSgroup == "4-0.8-0.5" &
iso_info$mayor])
iso_info$meanC1Sim[iso_info$DIEDSgroup == "4-0.8-0.5" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "4-0.8-0.5" & iso_info$mayor]*0.8
iso_info$varC1Sim[iso_info$DIEDSgroup == "4-0.8-0.5" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "4-0.8-0.5" & iso_info$mayor]*0.8^2
iso_info$meanC2Sim[iso_info$DIEDSgroup == "4-0.8-0.5" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "4-0.8-0.5" & iso_info$mayor]*(4*0.5)
iso_info$varC2Sim[iso_info$DIEDSgroup == "4-0.8-0.5" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "4-0.8-0.5" & iso_info$mayor]*(4*0.5)^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DIEDSgroup == "4-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "4-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*(1-0.8)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DIEDSgroup == "4-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.8)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "4-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.5)/(numb_iso-1)*4)
  varC2_sim<-iso_info$varC1[iso_info$DIEDSgroup == "4-0.8-0.5" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.5)/(numb_iso-1)*4)^2
  index<-which(iso_info$DIEDSgroup == "4-0.8-0.5" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]

# 4-0.8-0.3
altGene<-as.character(iso_info$gene_id[iso_info$DIEDSgroup == "4-0.8-0.3" &
iso_info$mayor])
iso_info$meanC1Sim[iso_info$DIEDSgroup == "4-0.8-0.3" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "4-0.8-0.3" & iso_info$mayor]*0.8
iso_info$varC1Sim[iso_info$DIEDSgroup == "4-0.8-0.3" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "4-0.8-0.3" & iso_info$mayor]*0.8^2
iso_info$meanC2Sim[iso_info$DIEDSgroup == "4-0.8-0.3" & iso_info$mayor]<-
iso_info$meanC1[iso_info$DIEDSgroup == "4-0.8-0.3" & iso_info$mayor]*(4*0.3)
iso_info$varC2Sim[iso_info$DIEDSgroup == "4-0.8-0.3" & iso_info$mayor]<-
iso_info$varC1[iso_info$DIEDSgroup == "4-0.8-0.3" & iso_info$mayor]*(4*0.3)^2
restinfo<-do.call(rbind,lapply(as.character(altGene), function(gen){
  numb_iso<-unique(iso_info$numb_iso[iso_info$DIEDSgroup == "4-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen])
  meanC1_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "4-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen]*(1-0.8)/(numb_iso-1)
  varC1_sim<-iso_info$varC1[iso_info$DIEDSgroup == "4-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.8)/(numb_iso-1))^2
  meanC2_sim<-iso_info$meanC1[iso_info$DIEDSgroup == "4-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.3)/(numb_iso-1)*4)
  varC2_sim<-iso_info$varC1[iso_info$DIEDSgroup == "4-0.8-0.3" &
  !iso_info$mayor & iso_info$gene_id == gen]*((1-0.3)/(numb_iso-1)*4)^2
  index<-which(iso_info$DIEDSgroup == "4-0.8-0.3" & !iso_info$mayor &
  iso_info$gene_id == gen)
  return(data.frame(index=index, meanC1_sim=meanC1_sim,  meanC2_sim=meanC2_sim,
  varC1_sim=varC1_sim,varC2_sim=varC2_sim))
}))
iso_info[restinfo$index,c("meanC1Sim", "meanC2Sim", "varC1Sim", "varC2Sim")]<-
restinfo[,-1]
```

```
## 4) Diferential gene expression

iso_info$meanC2Sim[iso_info$DE & iso_info$DEgroup %in% c(2,4)]<-
iso_info$meanC1[iso_info$DE & iso_info$DEgroup %in% c(2,4)] * as.numeric(
as.character(iso_info$DEgroup[iso_info$DE & iso_info$DEgroup %in% c(2,4)]))
iso_info$varC2Sim[iso_info$DE & iso_info$DEgroup %in% c(2,4)]<-
iso_info$varC1[iso_info$DE & iso_info$DEgroup %in% c(2,4)]*(as.numeric(
as.character(iso_info$DEgroup[iso_info$DE & iso_info$DEgroup %in% c(2,4)])))^2
iso_info$meanC1Sim[iso_info$DE & iso_info$DEgroup %in% c(0.25,0.5)]<-
iso_info$meanC1[iso_info$DE & iso_info$DEgroup %in% c(0.25,0.5)]/as.numeric(
as.character(iso_info$DEgroup[iso_info$DE & iso_info$DEgroup %in%
c(0.25,0.5)]))
iso_info$varC1Sim[iso_info$DE & iso_info$DEgroup %in% c(0.25,0.5)]<-
iso_info$varC1[iso_info$DE & iso_info$DEgroup %in% c(0.25,0.5)]/(as.numeric(
as.character(iso_info$DEgroup[iso_info$DE & iso_info$DEgroup %in%
c(0.25,0.5)])))^2

## Generating replicates values
# NB parameters
meansC1<-iso_info$meanC1Sim
meansC2<-iso_info$meanC2Sim
f.loess.C1<- loess(varC1 ~ mediasC1, data.frame(mediasC1=meansC1,
varC1=iso_info$varC1Sim), degree=2)
f.loess.C2<- loess(varC2 ~ mediasC2, data.frame(mediasC2=meansC2,
varC2=iso_info$varC2Sim), degree=2)
# variance prediction
var.predict1 <- predict(f.loess.C1, data.frame(mediasC1=meansC1))
var.predict2 <- predict(f.loess.C2, data.frame(mediasC2=meansC2))
size<-meansC1^2/(var.predict1+meansC1)
prob<-size/(size+meansC1)

numberReplicates<-10
for(nsim in 1:numberReplicates){
   setwd(paste("/path_to_simulation_scenario/sim", "nsim"))
   C1repeticiones10<-t(sapply(1:nrow(iso_info), function(iso){
     iso_cts_C1<-t(sapply(1:10, function(x){
        cond1<-rnbinom(n=4, size=size[iso], prob=prob[iso])
        return(c(cond1))
      }))
     cero_index_cts<-which(is.na(iso_cts_C1[,1]) | is.na(iso_cts_C1[,2]) |
     is.na(iso_cts_C1[,3]) | is.na(iso_cts_C1[,4]))
     iso_cts_C1[cero_index_cts,]<-0
     return(colMeans(iso_cts_C1))
   }))
```

```
iso_cts_C1<-as.data.frame(C1repeticiones10)
rownames(iso_cts_C1)<-iso_info$transcript_id
# Condition2
size<-meansC2^2/(var.predict2+meansC2)
prob<-size/(size+meansC2)
C2repeticiones10<-t(sapply(1:nrow(iso_info), function(iso){
 iso_cts_C2<-t(sapply(1:10, function(x){
   cond2<-rnbinom(n=4, size=size[iso], prob=prob[iso])
   return(c(cond2))
 }))
 cero_index_cts<-which(is.na(iso_cts_C2[,1]) | is.na(iso_cts_C2[,2]) |
 is.na(iso_cts_C2[,3]) | is.na(iso_cts_C2[,4]))
 iso_cts_C2[cero_index_cts,]<-0
 return(colMeans(iso_cts_C2))
}))
iso_cts_C2<-as.data.frame(C2repeticiones10)
rownames(iso_cts_C2)<-iso_info$transcript_id
iso_cm_sim<-data.frame(iso_cts_C1, iso_cts_C2)
names(iso_cm_sim)<-c("C1R1", "C1R2", "C1R3", "C1R4", "C2R1", "C2R2", "C2R3",
"C2R4")
rownames(iso_cm_sim)<-rownames(iso_cm)

# Generate files containing simulation profiles
transcript_ids<-as.character(iso_res[[1]]$transcript_id)
index<-match(transcript_ids, rownames(iso_cm_sim), nomatch=0)
iso_cm_sim_complete<-data.frame(matrix(0,nrow=length(transcript_ids),
ncol=ncol(iso_cm_sim)))
iso_cm_sim_complete[index!= 0,]<-iso_cm_sim[index[index !=0],]
rownames(iso_cm_sim_complete)<-transcript_ids
names(iso_cm_sim_complete)<-names(iso_cm_sim)
genes<-as.list(unique(as.character(iso_info$gene_id)))
sampleIndex<-c(1,2,3,4,11,15,17,20)
effect_length<-do.call(cbind,bplapply(sampleIndex, function(sample){
  return(iso_res[[sample]][,"effective_length"])
}, BPPARAM=MulticoreParam(10)))
effect_length<-as.data.frame(effect_length)
rownames(effect_length)<-rownames(iso_cm_sim_complete)
t_length<-do.call(cbind,bplapply(sampleIndex, function(sample){
  return(iso_res[[sample]][,"length"])
}, BPPARAM=MulticoreParam(10)))
t_length<-as.data.frame(t_length)
rownames(t_length)<-rownames(iso_cm_sim_complete)
scale_factors<-sapply(1:ncol(iso_cm_sim_complete), function(i){
  allratio<-sum(iso_cm_sim_complete[effect_length[,i] > 0,i] / effect_length[
effect_length[,i] > 0,i])
  return(allratio)
})

TPM<-do.call(rbind,bplapply(as.list(transcript_ids), function(iso){
   tpm<-iso_cm_sim_complete[rownames(iso_cm_sim_complete) ==
   iso,]/effect_length[rownames(effect_length) == iso,]
   if(any(is.na(tpm)) | any(tpm == Inf)){tpm[is.na(tpm) | tpm == Inf]<-0}
   tpm<-tpm*10^6/scale_factors
   return(tpm)
}, BPPARAM=MulticoreParam(18)))
sample_sizes<-colSums(iso_cm_sim)
FPKM<-sapply(1:ncol(iso_cm_sim_complete), function(sample){
   FPKM<-(iso_cm_sim_complete[,sample]) / (effect_length[,
   sample])*10^9/sample_sizes[sample]
   FPKM[is.na(FPKM) | FPKM == Inf]<-0
   return(FPKM)
  })
FPKM<-as.data.frame(FPKM)
names(FPKM)<-names(TPM)
rownames(FPKM)<-rownames(TPM)
index<-match(transcript_ids,g2t$transcript_id, nomatch=0)
g2t_ord<-g2t[index,]
# writing isoform files
sim_data<-bplapply(1:ncol(iso_cm_sim_complete), function(sample){
   sample_data<-data.frame(transcript_id=transcript_ids,
   gene_id=g2t_ord$gene_id,
   length=as.character(round(t_length[,sample],2)),effective_length=
   as.character(round(effect_length[,sample],2)),
   expected_count=as.character(round(iso_cm_sim_complete[,sample],2)),
   TPM=as.character(round(TPM[,sample],2)),FPKM=as.character(round(FPKM[,
   sample],2)),IsoPct=as.character(round(ratios_complete[,sample]*100,2)))
   return(sample_data)
}, BPPARAM=MulticoreParam(10))
names(sim_data)<-names(iso_res)[sampleIndex]
lapply(1:length(sim_data), function(sim){
   write.table(sim_data[[sim]], file=paste(names(sim_data)[sim],
   "_sim_iso.results", sep=""), row.names=FALSE, col.names=TRUE,dec=".",
   quote=FALSE, sep="\t")
  })
write.table(gene_info, "gene_info.tab", sep="\t", row.names=FALSE,
col.names=TRUE, quote=FALSE)
```

```
    write.table(iso_info, "iso_info.tab", sep="\t", row.names=FALSE,
    col.names=TRUE, quote=FALSE)
    save.image("simulation.RData", compress="xz")
}
```

APPENDIX III

```
# DIEAnalysis.R
library("DESeq2");library("NOISeq");library("EBSeq");library("limma");
library(edgeR)
# This script should be ran over each simulation of each scenario
setwd("/path_to_DIE_scenario_sim/")
g2t<-read.delim("gene2transc.txt", header=FALSE)
colnames(g2t)<-c("gene_id", "transcript_id")
# Preparing expression matrix
path_to_RSEM_sim_scenario
gene_files<-c("SRR057649.genes.results", "SRR057650.genes.results",
"SRR057651.genes.results", "SRR057652.genes.results",
"SRR057631.genes.results", "SRR057643.genes.results",
"SRR057645.genes.results", "SRR057648.genes.results")
gene_res<-lapply(gene_files, function(gene_f){
  return(read.delim(gene_f))
})
names(gene_res)<-sapply(gene_files,function(gene_f){
  return(strsplit(gene_f, split="[.]")[[1]][1])
})
gene_cm<-as.data.frame(do.call(cbind, lapply(1:length(gene_res), function(y){
  gene_sample<-gene_res[[y]]
  return(gene_sample$expected_count)
})))
rownames(gene_cm)<-gene_res[[1]]$gene_id
colnames(gene_cm)<-names(gene_res)
# isoforms
iso_files<- c("SRR057649.isoforms.results", "SRR057650.isoforms.results",
"SRR057651.isoforms.results", "SRR057652.isoforms.results",
"SRR057631.isoforms.results", "SRR057643.isoforms.results",
"SRR057645.isoforms.results","SRR057648.isoforms.results")
iso_res<-lapply(iso_files, function(iso_f){
  return(read.delim(iso_f))
})
names(iso_res)<-sapply(iso_files,function(iso_f){
  return(strsplit(iso_f, split="[.]")[[1]][1])
})
iso_cm<-as.data.frame(do.call(cbind, lapply(1:length(iso_res), function(x){
  iso_sample<-iso_res[[x]]
  return(iso_sample$expected_count)
})))
rownames(iso_cm)<-iso_res[[1]]$transcript_id
colnames(iso_cm)<-c(paste("C1R", 1:4, sep=""),paste("C2R",1:4, sep=""))
```

```
# filtering low quality genes. Only consider those that have counts in all
samples
index_g<-unique(do.call(c,sapply(1:nrow(gene_cm), function(x){
  if(any(gene_cm[x,1:8]==0) | any(gene_cm[x,9:16]==0) ){return(x)}else{
  return(NULL)}
  })))
gene_cm<-gene_cm[-index_g,]
g2t_ok<-g2t[(g2t$gene_id%in% rownames(gene_cm)),]
iso_cm<-iso_cm[rownames(iso_cm) %in% g2t_ok$transcript_id,]
genes<-sort(as.character(unique(g2t_ok$gene_id)))

# Only consider simulated genes
iso_info_sim<-read.delim("/path_to_simulation_scenario/sim/iso_info.tab",
stringsAsFactor=F)
gene_info_original<-read.delim("/path_to_simulation_scenario/sim/
gene_info.tab", header=T, stringsAsFactor=F)
sim_genes<-unique(iso_info_sim[, "gene_id"])
genes<-genes[genes%in% sim_genes]
gene_cm<-gene_cm[rownames(gene_cm) %in% genes,]
sim_iso<-iso_info_sim$transcript_id
iso_cm<-iso_cm[rownames(iso_cm) %in% sim_iso,]
rownames(gene_info_original)<-gene_info_original$gene_id
numb_iso<-gene_info_original[genes, "numb_iso"]
gene_info<-data.frame(gene_id=genes, numb_iso=numb_iso)
freq<-data.frame(table(gene_info$numb_iso[!gene_info$numb_iso ==1]),
accum=cumsum(table(gene_info$numb_iso[!gene_info$numb_iso
==1]))/sum(table(gene_info$numb_iso[!gene_info$numb_iso ==1])))
colnames(freq)<-c("Iso_num", "freq", "cum_rel_freq")
freq[,"score"]<-cut(freq[,"cum_rel_freq"], breaks=c(0,0.33,0.67,1),
include.lowest=FALSE, right=TRUE)
gene_info$group<-gene_info_original[genes, "group"]
all(gene_info$gene_id == rownames(gene_cm))
# [1] TRUE
gene_info$meanC1<-rowMeans(gene_cm[,1:4])
gene_info$meanC2<-rowMeans(gene_cm[,5:8])
gene_info$mean<-rowMeans(gene_cm)
gene_info$varC1<-apply(gene_cm[,1:4],1,var)
gene_info$varC2<-apply(gene_cm[,5:8],1,var)
gene_info$var<-apply(gene_cm,1,var)
iso_info<-merge(x=g2t_ok ,y=gene_info, by="gene_id", sort=FALSE)
iso_info<-iso_info[order(iso_info$gene_id,
iso_info$transcript_id),c(2,1,3:ncol(iso_info))]
all(iso_info$transcript_id == rownames(iso_cm))
# [1] TRUE
```

```
iso_info$iso_meanC1<-rowMeans(iso_cm[,1:4])
iso_info$iso_meanC2<-rowMeans(iso_cm[,5:8])
iso_info$iso_mean<-rowMeans(iso_cm)
iso_info$iso_varC1<-apply(iso_cm[,1:4],1,var)
iso_info$iso_varC2<-apply(iso_cm[,5:8],1,var)
iso_info$iso_var<-apply(iso_cm,1,var)
iso_info$ratiosC1<-iso_info$iso_meanC1/iso_info$meanC1
iso_info$ratiosC2<-iso_info$iso_meanC2/iso_info$meanC2
id_mayor<-iso_info_sim$transcript_id[iso_info_sim$mayor]
g2t_ok$mayor<-g2t_ok$transcript_id %in% id_mayor
iso_info$mayor<-iso_info$transcript_id %in% g2t_ok$transcript_id[ g2t_ok$mayor]

DIE_genes<-unique(iso_info_sim[iso_info_sim$DIE , "gene_id"])
DS_genes<-unique(iso_info_sim[iso_info_sim$DS, "gene_id"])
SDDIE_genes<-unique(iso_info_sim[iso_info_sim$DIEDS , "gene_id"])
DEgenes<-unique(iso_info_sim[iso_info_sim$DE, "gene_id"])
gene_info$DIE<-gene_info$gene_id %in% DIE_genes
gene_info$DS<-gene_info$gene_id %in% DS_genes
gene_info$DIEDS<-gene_info$gene_id %in% SDDIE_genes
gene_info$DE<-gene_info$gene_id %in% DEgenes
gene_info$DIEgroup<-factor(rep(0, nrow(gene_info)), levels=levels(as.factor(
gene_info_original$DIEgroup)))
idx<-match(gene_info$gene_id,gene_info_original$gene_id, nomatch=0)
gene_info$DIEgroup[gene_info$gene_id %in% gene_info_original$gene_id]<-
as.factor(gene_info_original$DIEgroup[idx])
gene_info$DSgroup<-factor(rep(0, nrow(gene_info)), levels=levels(as.factor(
gene_info_original$DSgroup)))
idx<-match(gene_info$gene_id,gene_info_original$gene_id, nomatch=0)
gene_info$DSgroup[gene_info$gene_id %in% gene_info_original$gene_id]<-
gene_info_original$DSgroup[idx]
gene_info$DIEDSgroup<-factor(rep(0, nrow(gene_info)), levels=levels(as.factor(
gene_info_original$DIEDSgroup)))
idx<-match(gene_info$gene_id,gene_info_original$gene_id, nomatch=0)
gene_info$DIEDSgroup[gene_info$gene_id %in% gene_info_original$gene_id]<-
gene_info_original$DIEDSgroup[idx]
gene_info$DEgroup<-factor(rep(0, nrow(gene_info)), levels=levels(as.factor(
gene_info_original$DEgroup)))
idx<-match(gene_info$gene_id,gene_info_original$gene_id, nomatch=0)
gene_info$DEgroup[gene_info$gene_id %in% gene_info_original$gene_id]<-
gene_info_original$DEgroup[idx]
iso_info<-merge(x=gene_info[,c(1,10:17)], y=iso_info, by="gene_id")
iso_info<-iso_info[,c(10,1,11:ncol(iso_info), 2:9)]
save(iso_info, file="iso_info_final_sim.RData", compress="xz")
save(iso_cm,file="iso_cm_sim.RData", compress="xz")
save(gene_info, file="gene_info_final_sim.RData", compress="xz")

# DIE analysis
conditions<-factor(c(rep("Normal",4), rep("Tumor",4)), levels=c("Normal",
"Tumor"))
isoNames<-rownames(iso_cm)
isoCPM<-sapply(1:ncol(iso_cm),function(x){iso_cm[,x]/sum(iso_cm[,x])*10^6})
isoCPM<-as.data.frame(isoCPM)
rownames(isoCPM)<-rownames(iso_cm)
colnames(isoCPM)<-colnames(iso_cm)
# Consider those isoforms expressed in at least one condition
isoCPM<-isoCPM[rowMeans(isoCPM[,1:4])>=4 | rowMeans(isoCPM[,5:8] >=4),]
iso_cm_expres<-iso_cm[rownames(iso_cm) %in% rownames(isoCPM),]
# save(iso_cm_expres, file="iso_cm_expres.RData")
iso_info_RSEM<-iso_info[iso_info$transcript_id %in% rownames(iso_cm_expres),]
# save(iso_info_RSEM, file="iso_info_express_sim.RData")
gene_info<-gene_info[gene_info$gene_id %in% unique(iso_info_RSEM$gene_id),]
# save(gene_info, file="gene_info_express_sim.RData")

##EBSeq
isoGeneNames<-read.delim(iso_files[[1]])[,1:2]
isoMat<-as.matrix(iso_cm_expres)
isoNames<-rownames(iso_cm_expres)
index<-match(isoGeneNames$transcript_id, isoNames)
index<-which(!is.na(index))
isoGeneNames<-as.character(isoGeneNames[index,"gene_id"])
# defining uncertainty groups
ngList<-GetNg(IsoformName=isoNames, GeneName=isoGeneNames, TrunThre=10)
isoNgTrun<-ngList$IsoformNgTrun #grupo al que pertenece
isoSizes<-MedianNorm(isoMat)
isoEBOut<-EBTest(Data=isoMat,
NgVector=isoNgTrun,Conditions=conditions,sizeFactors=isoSizes, maxround=8,
Qtrm=0.99, QtrmCut=0)
# The posterior probabilities of being DE are obtained is used equivalently to
FDR. Is obtained as follows, where PP is a matrix containing the posterior
isoPP<-GetPPMat(isoEBOut)
# The matrix PP contains two columns PPEE and PPDE, corresponding to the
posterior probabilities of being EE (equivalent expression) or DE (Differential
expression) for each gene. PP may be used to form an FDR-controlled list of DE
genes with a target FDR of 0.05 as follows:
isoDE<-rownames(isoPP)[which(isoPP[,"PPDE"]>=.95)]
save(isoEBOut,file="ebseq_res.RData")

## DESeq2
```

```
sample_data<-data.frame(sample=names(iso_cm_expres), condition=conditions)
y<-DESeqDataSetFromMatrix(countData=as.matrix(round(iso_cm_expres)),
colData=sample_data, design=formula(~condition, sample_data))
y<-estimateSizeFactors(y)
y<-estimateDispersions(y)
et_y_DES<-nbinomWaldTest(y)
DESeqres<-results(et_y_DES,c("condition","Tumor", "Normal"))
isoDESeq<-rownames(DESeqres[DESeqres$padj < 0.05 & !is.na(DESeqres$padj),])
save(et_y_DES, file="DESeqRes.RData")

## NOISeq
myfactors<-sample_data[,"condition", drop=FALSE]
mydata <- readData(data = (iso_cm_expres),factors=myfactors)
mynoiseqbio <- noiseqbio(mydata, norm = "tmm", k = 0, nclust=15,lc = 0,
conditions = myfactors, factor = "condition", r = 50, plot =FALSE, filter=0,
random.seed=123456789)
mynoiseqbio.deg <- degenes(mynoiseqbio, q=0.95)
DENoi<-rownames(mynoiseqbio.deg)
save(mynoiseqbio, file="noiseqRes.RData")

##Limma
dge <- DGEList(counts=iso_cm_expres)
dge <- calcNormFactors(dge)
design<-model.matrix(~condition, data=sample_data)
v <- voom(dge,design,plot=FALSE)
fit <- lmFit(v,design)
fit <- eBayes(fit)
limmaResults<-data.frame(baseMean=exp(fit$coefficients[,1]),
logFC=fit$coefficients[,2], pval=fit$p.value[,2])
limmaResults$padj<-p.adjust(limmaResults$pval, method="BH")
isoLimma<-rownames(limmaResults[limmaResults$padj < 0.05,])
save(fit, file="Limma.RData", compress="xz")

##Cufflinks
cuffresults<-read.delim("cufflinks_sim_scenario/isoform_exp.diff")
iso_track<-read.delim("cufflinks_sim_scenario isoforms.fpkm_tracking")
cuffresults_exp<-cuffresults[cuffresults$status == "OK",]
cuffresults_exp$test_id<-as.character(cuffresults_exp$test_id)
cuffresults_exp$gene<-as.character(cuffresults_exp$gene)
rownames(iso_track)<-iso_track$tracking_id
cuffresults_exp$nearest_ref_id<-as.character(iso_track[cuffresults_exp$test_id,
"nearest_ref_id"])
rownames(cuffresults_exp)<-cuffresults_exp$test_id
# there are some ensembl_ids mapping against more than one cuff_id
```

```
duplID<-unique(cuffresults_exp$nearest_ref_id[duplicated(
cuffresults_exp$nearest_ref_id)])
uniqueID<-unique(cuffresults_exp$nearest_ref_id[!duplicated(
cuffresults_exp$nearest_ref_id)])
uniqueID<-uniqueID[!(uniqueID %in% duplID)]
aux<-sapply(1:length(duplID),function(id){
  dupl<-cuffresults_exp[cuffresults_exp$nearest_ref_id == duplID [id], ,
  drop=FALSE]
  if(any(dupl$significant == "yes")){
    index<-which(dupl$significant == "yes")[1]
  }else{index<-which.min(abs(dupl$q_value))[1]}
  return(as.character(dupl$test_id[index]))
})
cuffresults_exp<-
cuffresults_exp[c(cuffresults_exp$test_id[cuffresults_exp$nearest_ref_id %in%
uniqueID], aux),]
iso_info_cuff<-iso_info[iso_info$transcript_id %in%
cuffresults_exp$nearest_ref_id, ]
cuffresults_exp<-cuffresults_exp[cuffresults_exp$nearest_ref_id %in%
iso_info_cuff$transcript_id, ]
DEcuff<-unique(cuffresults_exp$nearest_ref_id[cuffresults_exp$q_value <=0.05])
save(cuffresults_exp, file="cuffdiffIsoSim.RData", compress="xz")

save.image("all.RData", compress="xz")
```

APPENDIX IV

```
# DSAnalysis.R
conditions<-factor(c(rep("Normal",4),rep("Tumor",4)), levels=c("Normal",
"Tumor"))

library(SplicingCompass); library(DEXSeq); library(limma)
setwd("/path_to_DS_scenario_sim/")
expInf<-new("ExperimentInfo")
expInf<-setDescription(expInf,"NormalVsTumor")
expInf<-setGroupInfo(expInf, groupName1="Normal", sampleNumsGroup1=1:4,
groupName2="Tumor", sampleNumsGroup2=5:8)
covBedCountFilesNormal<-c(
"/path_to_quantification_covBed_sim_scenario/SRR057649.covBed.counts",
"/path_to_quantification_covBed_sim_scenario/SRR057650.covBed.counts",
"/path_to_quantification_covBed_sim_scenario/SRR057651.covBed.counts",
"/path_to_quantification_covBed_sim_scenario/SRR057652.covBed.counts")
covBedCountFilesTumor<-
c("/path_to_quantification_covBed_sim_scenario/SRR057631.covBed.counts",
"/path_to_quantification_covBed_sim_scenario/SRR057643.covBed.counts",
"/path_to_quantification_covBed_sim_scenario/SRR057645.covBed.counts",
"/path_to_quantification_covBed_sim_scenario/SRR057648.covBed.counts")
expInf<-setCovBedCountFiles(expInf, c(covBedCountFilesTumor,
covBedCountFilesNormal))
junctionBedFilesNormal<-
c("/path_to_alignment_genome_sim_scenario/SRR057649/junctions.bed",
"/path_to_alignment_genome_sim_scenario/SRR057650/junctions.bed",
"/path_to_alignment_genome_sim_scenario/SRR057651/junctions.bed",
"/path_to_alignment_genome_sim_scenario/SRR057652/junctions.bed")
junctionBedFilesTumor<-c(
c("/path_to_alignment_genome_sim_scenario/SRR057631/junctions.bed",
"/path_to_alignment_genome_sim_scenario/SRR057643/junctions.bed",
"/path_to_alignment_genome_sim_scenario/SRR057645/junctions.bed",
"/path_to_alignment_genome_sim_scenario/SRR057648/junctions.bed")
expInf<-setJunctionBedFiles(expInf, c(junctionBedFilesTumor,
junctionBedFilesNormal))
expInf<-setReferenceAnnotation(expInf,
“/path_to_annotation_files/flattened.splcmp.gtf”)
referenceAnnotationFormat<-list(IDFieldName="geneSymbol",idValSep=" ")
expInf<-setReferenceAnnotationFormat(expInf,referenceAnnotationFormat)
checkExperimentInfo(expInf)
## Constructing an object of class CountTable
mycountTable<-new("CountTable")
mycountTable<-setExperimentInfo(mycountTable, expInf)
```

```
mycountTable<-constructCountTable(mycountTable,nCores=20, printDotPerGene=TRUE)
sc<-new("SplicingCompass")
sc<-constructSplicingCompass(sc, mycountTable, minOverallJunctionReadSupport=5,
nCores=20)

# obataining significant DE genes
sc<-initSigGenesFromResults(sc, adjusted=TRUE, threshold=0.05)
sigGenes<-getSignificantGeneSymbols(sc)
# obtaining a data frame with tested genes and correspoonding p-values
resTab<-getResultTable(sc)
resTab<-resTab[resTab$gene_id %in% iso_info$gene_id,]
# tested gene ID
genesTested<-getAllTestedGenes(sc)

iso_info_SC<-iso_info[iso_info$gene_id %in% genesTested,]
sigGenes<-getSignificantGeneSymbols(sc)
sigGenes<-sigGenes[sigGenes %in% iso_info_SC$gene_id]
save(resTab, file="SCresultTableNOv.RData")
save(sc, file="SCobjectNOv.RData")
save(mycountTable, file="SCCountTableNov.RData")

#DEXSeq
countfiles<-paste("path_to_quantification_DEXSeq_sim_scenario/htseq_sim_",
paste("SRR0576", c(49:52, 31, 43, 45, 48),".htseq.counts", sep="")
# building design matrix
sample_data<-data.frame(condition= conditions, levels=c("Normal","Tumor")))
design<-~sample+exon+condition:exon
row.names(sample_data)<-c(paste("C1R", 1:8, sep=""), paste("C2R", 1:8, sep=""))
# build dexseq count matrix from HTseq output
count_matrix<-DEXSeqDataSetFromHTSeq(countfiles, sample_data,design,
flattenedfile="/path_to_annotation_files/flattened.dexseq.gtf")
count_matrix<-estimateSizeFactors(count_matrix)
count_matrix<-estimateDispersions(count_matrix, maxit=500,
BPPARAM=MulticoreParam(workers=18))
fullModel<- ~sample + exon + condition:exon
reducedModel<- ~sample + exon
count_matrix<-testForDEU(count_matrix, fullModel=fullModel,
reducedModel=reducedModel, BPPARAM=MulticoreParam(workers=20))
count_matrix<-estimateExonFoldChanges(count_matrix, fitExpToVar="condition",
BPPARAM=MulticoreParam(workers=20), denominator="Normal")
myresults<-DEXSeqResults( count_matrix )
perGeneQ<-perGeneQValue(myresults)
myresultsDF<-as.data.frame(myresults)
myresultsDF<-myresultsDF[!is.na(myresultsDF$padj) ,]
```

```
myresultsDF$qvalGene<-do.call(c, lapply(1:nrow(myresultsDF), function(i){
return(perGeneQ[names(perGeneQ) == myresultsDF$groupID[i]])
}))
myresultsDF<-myresultsDF[myresultsDF[,"groupID"]%in%unique(iso_info$gene_id), ]
iso_info_dexseq<-iso_info[iso_info$gene_id %in%
unique(myresultsDF[,"groupID"]), ]
DEXGenes<-unique(myresultsDF[myresultsDF$qvalGene < 0.05,"groupID"])
save(count_matrix, file="DEXSeqCountMatrixSim.RData")

#LimmaDS
cm<-counts(count_matrix)[,1:8]
cm2<-as.data.frame(rowRanges(count_matrix))
geneInfo<-cm2[,c(1:7)]
y.all <- DGEList(counts=cm, genes=geneInfo)
isexpr <- rowSums(cpm(y.all) > 1) >=3
y <- y.all[isexpr,,keep.lib.sizes=FALSE]
save(y , file="expressCMNover.RData", compress="xz")
y <- calcNormFactors(y)
design <- model.matrix(~ conditions)
v <- voom(y,design,plot=FALSE)
fit <- lmFit(v, design)
fit.de <- eBayes(fit, robust=TRUE)
limmaResults<-data.frame(gene=fit.de$genes,
baseMean=exp(fit.de$coefficients[,1]), logFC=fit.de$coefficients[,2],
pval=fit.de$p.value[,2])
limmaResults$padj<-p.adjust(limmaResults$pval, method="BH")
ex <- diffSplice(fit[,"conditionTumor"], geneid = "groupID", exonid = "start")
DSRes<-topSplice(ex, test="simes", n=length(ex))
iso_info_limma<-iso_info[iso_info$gene_id %in% DSRes[,"groupID"],]
DSRes<-DSRes[DSRes$groupID %in% unique(iso_info$gene_id ),]
DELimma<-DSRes[DSRes$FDR < 0.05,"groupID"]
save(ex, file="limmaDSNOver.RData", compress="xz")
save(fit.de, file="fitdeLimmaNOver.RData", compress="xz")
save(DSRes, file="LimmaDSRes.RData", compress="xz")

##CufflinksDS
cuffresults<-read.delim("cufflinks_sim_scenario/splicing.diff")
cuffresults_exp<-cuffresults[cuffresults$status == "OK",]
cuffresults_exp$test_id<-as.character(cuffresults_exp$test_id)
cuffresults_exp$gene<-as.character(cuffresults_exp$gene)
rownames(cuffresults_exp)<-cuffresults_exp$test_id
duplIDs<-duplicated(paste(cuffresults_exp$gene, cuffresults_exp$locus,
sep="_"))
```

```
duplIDs<-data.frame(test_id=cuffresults_exp$test_id, gene=cuffresults_exp$gene,
locus=cuffresults_exp$locus, duplicados=duplicados)
duplIDs<- duplIDs[duplIDs$duplIDs,]
dim(duplIDs)
# [1] 1968    4
uniqID<-cuffresults_exp$test_id[!(paste(cuffresults_exp$gene,
cuffresults_exp$locus, sep="_") %in% paste(duplIDs$gene, duplIDs$locus,
sep="_"))]
aux<-sapply(1:nrow(duplIDs),function(id){
  dupl<-cuffresults_exp[(cuffresults_exp$locus == duplIDs[id, "locus" ]) &
  cuffresults_exp$gene == duplIDs[id, "gene"],,drop=FALSE]
  if(any(dupl$significant == "yes")){
    id<-dupl$test_id[which(dupl$significant == "yes")[1]]
  }else{id<-dupl$test_id[which.min(abs(dupl$q_value))[1]]}
  return(as.character(id))
})
cuffresults_exp<-cuffresults_exp[cuffresults_exp$test_id %in% c(uniqID, aux),]

# convert gene name to ensembl
conversion_info<-
read.delim("~/prostate3case/simulation/DIE_results/moreRestrictive/gene_names_a
nd_entrez_interest2.tab", header=TRUE)
rownames(conversion_info)<-conversion_info$gene_name
cuffresults_exp<-cuffresults_exp[cuffresults_exp$gene %in%
cuffresults_exp$gene[(cuffresults_exp$gene %in% rownames(conversion_info))] ,]
cuffresults_exp$gene_id<-
as.character(conversion_info[cuffresults_exp$gene,"gene_id"])
table(cuffresults_exp$significant == "yes")
DEcuff<-unique(cuffresults_exp$gene_id[cuffresults_exp$q_value <=0.05])
cuffresults_exp<-cuffresults_exp[cuffresults_exp$gene_id %in%
unique(iso_info$gene_id),]
iso_info_cuff<-iso_info[iso_info$gene_id %in% cuffresults_exp$gene_id,]

save(cuffresults_exp, file="cuffresults_expDS.RData", compress="xz")

sessionInfo()
# R version 3.2.5 (2016-04-14)
# Platform: x86_64-pc-linux-gnu (64-bit)
# Running under: Ubuntu 14.04.5 LTS
#
# locale:
#  [1] LC_CTYPE=en_US.UTF-8    LC_NUMERIC=C            LC_TIME=C
#  [4] LC_COLLATE=en_US.UTF-8  LC_MONETARY=C           LC_MESSAGES=en_US.UTF-8
#  [7] LC_PAPER=C              LC_NAME=C               LC_ADDRESS=C
# [10] LC_TELEPHONE=C          LC_MEASUREMENT=C        LC_IDENTIFICATION=C
#
# attached base packages:
#  [1] splines   stats4    parallel  stats     graphics  grDevices utils
#  [8] datasets  methods   base
#
# other attached packages:
#  [1] edgeR_3.12.1               EBSeq_1.10.0
#  [3] testthat_1.0.2             gplots_3.0.1
#  [5] blockmodeling_0.1.8        NOISeq_2.14.1
#  [7] Matrix_1.2-7.1             limma_3.26.9
#  [9] DEXSeq_1.16.10             DESeq2_1.10.1
# [11] RcppArmadillo_0.7.400.2.0  Rcpp_0.12.7
# [13] SummarizedExperiment_1.0.2 GenomicRanges_1.22.4
# [15] GenomeInfoDb_1.6.3         IRanges_2.4.8
# [17] S4Vectors_0.8.11           Biobase_2.30.0
# [19] BiocGenerics_0.16.1        SplicingCompass_1.0.1
# [21] BiocParallel_1.4.3         gridExtra_2.2.1
# [23] ggplot2_2.1.0
#
# loaded via a namespace (and not attached):
#  [1] locfit_1.5-9.1       lattice_0.20-34      gtools_3.5.0
#  [4] Rsamtools_1.22.0     Biostrings_2.38.4    digest_0.6.10
#  [7] R6_2.1.3             plyr_1.8.4           chron_2.3-47
# [10] futile.options_1.0.0 acepack_1.3-3.3      RSQLite_1.0.0
# [13] zlibbioc_1.16.0      gdata_2.17.0         data.table_1.9.6
# [16] annotate_1.48.0      rpart_4.1-10         labeling_0.3
# [19] statmod_1.4.26       geneplotter_1.48.0   stringr_1.1.0
# [22] foreign_0.8-67       RCurl_1.95-4.8       biomaRt_2.26.1
# [25] munsell_0.4.3        nnet_7.3-12          Hmisc_3.17-4
# [28] XML_3.98-1.4         crayon_1.3.2         bitops_1.0-6
# [31] grid_3.2.5           xtable_1.8-2         gtable_0.2.0
# [34] DBI_0.5-1            magrittr_1.5         scales_0.4.0
# [37] KernSmooth_2.23-15   stringi_1.1.2        XVector_0.10.0
# [40] hwriter_1.3.2        genefilter_1.52.1    latticeExtra_0.6-28
# [43] futile.logger_1.4.3  Formula_1.2-1        lambda.r_1.1.9
# [46] RColorBrewer_1.1-2   tools_3.2.5          survival_2.39-5
# [49] AnnotationDbi_1.32.3 colorspace_1.2-6     cluster_2.0.4
# [52] caTools_1.17.1
```