ASpli: An integrative R package for analysing alternative splicing using RNAseq

Estefania Mancini, Marcelo Yanovsky, Ariel Chernomoretz

October 17, 2016

Contents

1 Introduction

Alternative splicing (AS) is a common mechanism of post-transcriptional gene regulation in eukaryotic organisms that expands the functional and regulatory diversity of a single gene by generating multiple mRNA isoforms that encode structurally and functionally distinct proteins. The development of novel high-throughput sequencing methods for RNA (RNA-Seq) has provided a powerful means of studying AS under multiple conditions and in a genome-wide manner [**?**]. However, using RNA-seq to study changes in AS under different experimental conditions is not trivial. In this vignette, we describe how to use ASpli, an integrative and user-friendly R package that facilitates the analysis of changes in both annotated and novel AS events. This package combines statistical information from exon, intron, and splice junction differential usage (p-value, FDR), with information from splice junction reads to calculate differences in the percentage of exon inclusion (Δ PSI) and intron retention (ΔPIR). The proposed methodology reliably reflect the magnitude of changes in the relative abundance of different annotated and novel AS events. This method can be used to analyze both simple and complex experimental designs involving multiple experimental conditions

2 Getting started

BAM files of each sample and gene annotation are mandatory. Make sure all files use the same coordinate system. It is a good practice to organize all required files (or symbolic links of them) together into the working directory.

2.1 Installation

ASpli is available at bioconductor.org and can be downloaded using `biocLite()`:

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

```
> library(ASpli)
```

`biocLite()` will take care of installing all the packages that ASpli depends on e.g. edgeR, DEXSeq, GenomicFeatures, GenomicRanges, GenomicAlignments, Gviz, and more...

2.2 Building a TxDb of your genome

Gene annotation is handled using TxDb objects. The building of a TxDb object might be time consuming, depending on the size of your genome. If a TxDb has been already created, we strongly recommend saving it as a sqlite file and reloading it any time the analysis is run. The GenomicFeatures package provides functions to create TxDb objects based on data downloaded from UCSC Genome Bioinformatics, BioMart or gff/gtf files. See "Creating New TxDb Objects or Packages" in the GenomicFeatures vignette for more details. In this example, a TxDb is built from a gtf file:

```
> TxDb <- makeTxDbFromGFF(
   file="genes.gtf",
   format="gtf")
```

3 Running ASpli

ASpli is divided in four independent steps or modules (Figure 1) :

1. Feature extraction: binGenome()
2. Read counting readCounts()
3. Alternative splicing discovery using junctions AsDiscover()
4. DE and DU estimation DUreport()

At each step it is possible to export results in a friendly manner. See Section "3.5 Outputs and results" for more details.

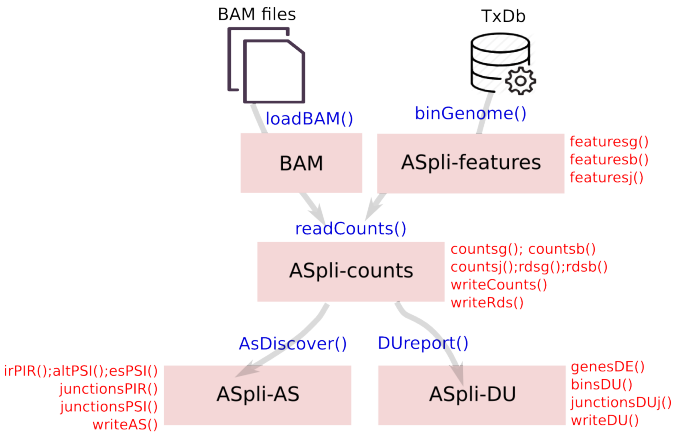


Figure 1: Objects (squares) and methods (in blue) in ASpli package. Accesors and write methods are in red.

3.1 To BIN or not to BIN

Sub-genic features such as exons and introns are analyzed using existing annotations by splitting the genome into non-overlapping features called bins, as described previously for DEXSeq [**?**]. Exon and intron coordinates are then extracted from the annotation, but only those from multi-exonic genes are saved for further evaluation. When more than one isoform exists, some exons and introns will overlap. Exons and introns are then subdivided into new features called exon and intron bins, and are then classified into exclusively exonic bins, exclusively intronic bins, or alternative splicing (AS) bins (See Figure 2). Each AS bin is then labeled as follows, according to the type of AS event it represents:

- **ES** (exon skipping)
- **IR** (intron retention)
- **ALt5'SS** alternative five prime splicing site
- **ALt3'SS** alternative three prime splicing site

- "*" (ES*, IR*, AltSS*) for bins involved in more than one AS event type
- **external**: beginning or end of a transcript

Annotated junctions from all the transcripts are also reported. Junction coordinates are defined as the last position of the five prime exon (donor position) and first position of the three prime exon (acceptor)

Subgenic features are labeled as follow (hypothetical GeneAAA):

- **GeneAAA:E001**: defines first exonic bin
- **GeneAAA:I001**: defines first intronic bin
- **GeneAAA:Io001**: defines first original intron before being divided
- **GeneAAA:J001**: defines first junction

Bins and junctions are labeled always in 5' to 3' sense of the plus strand. This implies that bins and junctions with lower numbering are always closer to the 5' of that strand.

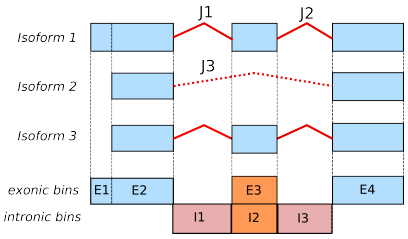


Figure 2: Scheme of resulting bins from a gene with two hypothetical transcripts. Those bins that are exonic or intronic at the same time are named *AS bins*. The junctions repertoire is also shown.

The `binGenome` method allows us to extract genome coordinates at the gene, exon, intron and junction level and returns an object of class `ASpliFeatures`. Then, features cordinates could be extractd using their accesors:

```
> features <- binGenome(TxDb)
> GenesCoord <- featuresg(features)
> BinsCoord <- featuresb(features)
> JunctionsCoord <- featuresj(features)
```

3.2 Read Counting

3.2.1 Mapping file and bam loading

Sample names, full path of BAM files and experimental factors should be specified in a dataframe (i.e. `targets`) under the header: `bam` and `condition`, and sample names as rownames. For example: for a one factor experiment with 2 replicates for each condition (Control and Mutant):

```
> bamFiles <- c("Ct1.bam", "Ct2.bam", "Mut1.bam","Mut2.bam" )
> targets <- data.frame(bam=bamFiles,
                        condition=rep(c("CT", "Mut"), each=2),
                        row.names=sub("\\.bam$", "", bamFiles))
```

More sofisticated designs, should be specified as follow, For example: for a two factor experiment with 2 replicates for each condition (Control, Mutant, Time1, Time2):

```
> bkg <- rep(c("CT", "Mut"), each=4)
> time <- rep(rep(c("T1", "T2"), each=2),2)
> bamFiles <- c("Ct_time1_rep1.bam", "Ct_time1_rep2.bam",
          "Ct_time2_rep1.bam", "Ct_time2_rep2.bam",
          "Mut_time1_rep1.bam","Mut_time1_rep2.bam",
          "Mut_time2_rep1.bam","Mut_time2_rep2.bam")
```

```
> targets_2 <- data.frame(bam=bamFiles,
                          condition=paste(bkg, time, sep="."),
                          row.names=sub("\\.bam$", "", bamFiles))
```

Using the `targets` data.frame, BAM files are loaded and stored as read alignments in an object of type list.

```
> bam <- loadBAM(targets)
```

3.2.2 Overlap features and read alignments

Using the method `readCounts`, read alignments are overlaid on features. Results are stored into an `ASpliCounts` object. Read densities (number of reads / feature length) are calculated for genes and bins. Count tables and read densities tables are produced at different genomic levels, including genes, bins, junctions, and intron flanking regions.

```
> counts <- readCounts(features, bam, l=100L, maxISize=5000)
```

Parameters:

- `l`: Read length of sequenced library
- `maxISize`: maximum intron expected size. Junctions longer than this size will be dicarded. [**?**]

Count and read density tables could be extracted using accesors. Also, it is possible to export count and read densities tables in a tidy manner.

```
> GeneCounts <- countsg(counts)
> GeneRd <- rdsg(counts)
> BinCounts <- countsb(counts)
> BinRd <- rdsb(counts)
> JunctionCounts <- countsj(counts)
> e1iCounts <- countse1i(counts)
> ie2Counts <- countsie2(counts)
> writeCounts(counts=counts, output.dir = "example")
> writeRds(counts=counts, output.dir = "example")
```

Some key features:

- **E1I and IE2 counts**: Every intron is considered as a potential retained intron. Analysis of putative IR events consider adjacent 5' and 3' exons (E1 and E2, respectively) and the intron itself (I). Following [**?**], two retention regions E1I (connecting exon E1 and I) and IE2 (connecting the intron and exon 2) and one constitutive (i.e., no retention) junction, E1E2 (connecting exons 1 and 2) are defined. The read length of sequenced library (`l`) is used for the definition of those new artifitial ranges:
  - E1I: intron start-(l-minAnchor)- intron start + (l-minAnchor)
  - IE2: intron end-(l-minAnchor)- intron end + (l-minAnchor)
  - `minAnchor` is 10% of read length by default (parameter `minAnchor`)

  Only those reads which minimum overlap *l* and without gap in this interval are considered. Accordingly, only sequences aligned within those two exons/introns are counted.
- Number of reads by gene are computed summarizing the reads of the constitutive exon bins of each gene.
- In order to estimate read densities, an effective length (sum of the length of exonic bins (constitutive and alternative of their corresponding gene) is considered
- The minimum overlap lenght between read and feature is 1. Hence, some reads might overlap to more than one bin and could be counted several times.
- Junctions: They are extracted from BAM files. They are defined as those reads aligned skipping region from the reference (N operation in the CIGAR). They are essential for alternative splicing event quantification and discovery.

3.3 Alternative splicing analysis and discovery using junction

Using the obtained count tables at bin and junction level it is possible to get an integrative view of the AS events under analysis. The function `AsDiscover` will produce an `ASpliAS` object containing several comprehensive tables, which could be accesed through their corresponding accesors. It is necessary to specify pair of comparison:

```
> pair <- c("CT","KD")
> as <- AsDiscover(counts, targets, features, bam, l=100L, pair=pair)
```

The analysis will consider junctions that:

- Are completely included into a unique gene (i.e. they are not multiple hit)
- Have more than a minimum number of reads supporting them (parameter `threshold, default=5`)

To provide an integrative view of the AS events being analyzed, splice junction information is used to analyze differential bin usage. PSI (percent of inclusion) and PIR (percent of intron retention) metrics, which are extensively used to quantify AS [**?**], are calculated. (see Figure 3) In the case of exonic bins, PSI values are computed using junctions that share start or end positions and fully span the exon. In the case of intronic bins, the PIR metric is computed using junctions that share start and end with each intronic bin, and read counts in regions spanning the intron/exon boundary. This information allows the user to obtain reliable information on the relative abundance of the AS event being evaluated. Both metrics strongly enrich the count-centric analysis and provide independent experimental support for the existence of novel AS events, based on the identification of corresponding changes in nearby splice junctions.
For each experimental junction identified, it is also reported if it is new and which bins are spanned. In addition, it is stated if the junction is completely included in an annotated bin, which would indicate that the AS event is a possible exintron [**?**].

In addition, it is also possible to access and export tables:

```
> irPIR <- irPIR(as)
> altPSI <- altPSI(as)
> esPSI <- esPSI(as)
> junctionsPIR <- junctionsPIR(as)
> junctionsPSI <- junctionsPSI(as)
> writeAS(as=as, output.dir="example")
```

New splicing events discovery
ASpli allows novel AS events to be identified based on the splice junction repertoire. A novel AS event is identified whenever a novel splice junction that shares its beginning or end with another splice junction is discovered. When a novel AS event is identified using the splice junction repertoire, the PSI metric is calculated and reported. This ability to detect novel AS events and to estimate the magnitude of the changes in the usage of these AS events under different experimental conditions are original functions of the package.

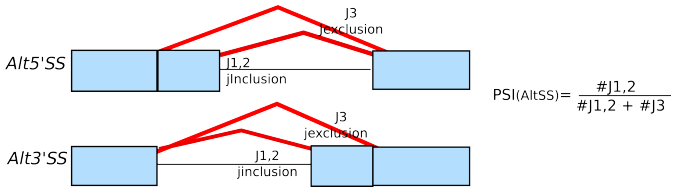
3.4 Estimating differential gene expression and bin/junction usage

Using the generated counting tables it is possible to estimate the differential usage (DU) of bins and junctions, and the differential expression at gene level (DE) using any analysis methodology of choice. For the sake of user convinience, we included two wrapper functions, `DUreport` and `DUreport_DEXseq`, that allow to perform such analysis using any of two widespread used methodologies: edgeR and DEXseq, respectively.

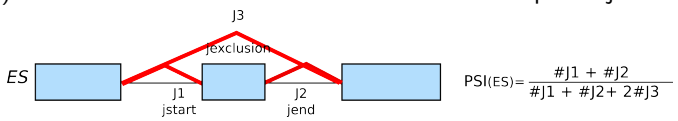
DUreport function performs statistical tests using a count centric approach. Results are stored into an `ASpliDU` object. Individual tables can be accesed using accesors and exported using `writeDU` function.

It is necessary to specify group and pair of comparison:

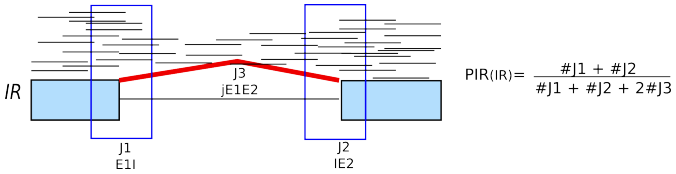
```
> pair <- c("CT","KD")
> group <- c(rep("CT", 4),rep("KD", 4))
> du <- DUreport(counts, targets, pair, group)
```



(a) PSI metric for AltSS estimation and its relationship with junctions



(b) PSI metric for ES estimation and its relationship with junctions



(c) PIR metric for IR estimation and its relationship with junctions

Figure 3: PSI and PIR metrics estimation and their relationship with junctions

3.4.1 Differential gene expression (DE)

Genes are considered expressed if they received in average (parameter `minGenReads`, default=10) more than a minimum number of reads and if average read density per condition is above a threshold in any condition (parameter `minRds`, default=0.05). Differential gene expression is estimated using edgeR [**?**] package. **logFC**, **pvalue**, and **adjusted pvalue** by false discovery rate [**?**] are reported.

3.4.2 Differential junctions and bin usage (DU)

In order to select informative events the following criteria are adopted:

- **Junctions**: junctions from expressed genes will be taken into account if they have more reads than a threshold supporting them (parameter `threshold`, default=5) in one of the conditions.

Where differential junction and bin usage are estimated using the statistical method proposed in the edgeR package, external bins are not included in the analysis (parameter `ignoreExternal`=TRUE, by default). and bin and junction counts are corrected as follow:

Given a Bin_i belonging to a $Gene_A$ in condition 1, Bin_i counts (G_A:B_i) are divided by $Gene_A$ counts in condition 1 and multiplied by the gene count average observed along all conditions $\tilde{G}_A$

average *Gene counts* in all conditions:

$$CountBinAdjusted = (\frac{G_A : B_i}{G_A})_1 * \tilde{G}_A$$

As already mentioned it is also possible to use DEXSeq for differential usage estimation, without any correction:

```
> du_DEXSeq <- DUreport_DEXSeq(counts, targets, pair, group)
```

In general, we have found a high overlap between both methods. (See example below, in Section 4 "Example data" analysis.)

Output tables can be accessed and exported easily:

```
> writeDU(du, output.dir="example")
> genesde <- genesDE(du)
> binsdu <- binsDU(du)
> junctionsdu <- junctionsDU(du)
```

3.5 Output and results

At each module, results are stored in ASpliObjects. Self-explanatory tables can be exported at each step of the analysis. Using write functions, it is possible to export tab delimited tables in a features-level output folder, as was detailed before:

```
> writeCounts(counts, "example_counts")
> writeRds(counts, "example_rds")
> writeDU(du, output.dir="example_du");
> writeAS(as=as, output.dir="example_as");
> writeAll(counts=counts, du=du, as=as, output.dir="example_all")
```

Information summary reported in tables:

Genomic metadata:

- Common info:
  - **locus_overlap**: whether overlap exists with another gene. Genes with the same coordinates are discarded and only one is kept (first one alphabetically).
  - **symbol**: common name
  - **gene_coordinates**: genomic coordinates in a shrink mode
  - **start, end** and **length**: genomic coordinate of corresponding feature
- Gene exclusive
  - **effective_length**: sum of length of exons bins of the gene (constitutive and alternative)
- Bin exclusive (exons, introns, e12, ie2 tables):
  - **feature**: levels are E (exon), I (intron), Io (original intron)
  - **event**: according to our classification, bins are classified into ES, IR, ALt5'SS, ALt3'SS, external. Bins tagged with * means this AS bin is involved simultaneously in more than one AS event type mean they came from
  - **locus**: gene coordinates
- Junction exclusive:
  - **junction**: if exist, name of existing junction
  - **gene**: if junction is inside an existing gene (within)
  - **strand**: corresponding strand of gene match
  - **multipleHit**: if junction matchs multiple locus
  - **bin_spanned**: bins spanned by junction
  - **j_within_bin**: if junction is included into a bin. This information is useful for the analysis of possible exintrons.

Junctions metadata:

The following junction dependant tables with their corresponding content are available (see Figure 3 a,b,c):

- **intron.pir**: event, e1i counts (J1), ie1 counts (J2), j_within (J3), PIR by condition. J1, J2, J3 are the sum of junctions (J1, J2, J3) by condition.

$$PIR = \frac{J1 + J2}{2 * J3 + (J1 + J2)}$$

- **exon.altPSI**: event, J1 (start), J2 (end), J3 (exclusion), PSI. J1, J2, J3 are the sum of junctions (J1, J2, J3) by condition.

$$PSI(AltSS) = \frac{J1,2}{J1,2 + J3)}$$

- **exon.altES**: event, J1 (start), J2 (end), J3 (exclusion), PSI. J1, J2, J3 are the sum of junctions (J1, J2, J3) by condition.

$$PSI(AltES) = \frac{J1 + J2}{J1 + J2 + 2 * J3)}$$

- **junctions.pir**: This table is similar to **intron.pir** table. Contains PIR metric for each experimental junction using e1i and ie2 counts. Exclusion junction is the junction itself. This table helps to discover new introns as well as new retention events. Original data:
  - hitIntron: compares junction with annotated introns
  - hitIntronEvent: reports if the annotated intron is alternative or not
- **junctions.psi**: Given a junction, we can analyze if it shares start, end or both with another junction. If so, it indicates the existance of an alternative splicing event. Using strand information it is possible to classify those junctions into ALt5'SS, ALt3'SS or ES. Ratio between them along samples is reported.

3.6 Plots

Inspection of coverage plots is a good practice for the interpretation of analytical results. After selection of AS candidates it is possible to plot the results in a genome browser manner highlighting alternatively spliced regions using a simple function `plotTopTags`.

- It is neccesary to build an small dataframe (`auxdf`) with the columns: `event, gene_coordinates, start of bin, end of bin` and bin names as rownames.
- In `targetsPlot` object you should specify which BAM files you will plot, in which colors and which sample names you want to display.

```
> bamFiles <- c("CT.bam","KD.bam")
> targetsPlot <- data.frame(bam=bamFiles,
                            sample=sub("\\.bam$", "", bamFiles),
                            color=c("blue", "black"),
                            stringsAsFactors=FALSE)
```

- You have to put `BAI` files of the corresponding BAM ones in the same folder and provide a `TxDb` object to the function.

And finally, you can plot:

```
> plotTopTags(auxdf,
              TxDb,
              targetsPlot,
              output.dir="testPlots")
```

4 Example data

It is possible to run a demo of `ASpli` using public RNA-seq data available via Bioconductor. It requieres installing of `RNAseqData.HNRNPC.bam.chr14` package and loading an small TranscriptDb of Human Genome only for chromosome 14. This subset of reads is part of an experiment intended to analyze splicing factors and their relationship with exonization of Alu elements [**?**]. We load the package and the genome:

```
> library(RNAseqData.HNRNPC.bam.chr14)
```

In this case we are loading a genome `TranscriptDb` already created and available in the `ASpli` package for the Homo sapiens chromosome 14:

```
> chr14 <- system.file("extdata","chr14.sqlite", package="ASpli")
> genome <- loadDb(chr14)
```

And now we are ready to start the complete analysis.

```
> features <- binGenome(genome)
```

Create targets object using the information of the available files:

```
> targets <- data.frame(bam=RNAseqData.HNRNPC.bam.chr14_BAMFILES,
                        condition=c(rep("CT",4),rep("KD",4)))
>
```

Load bam files:

```
> bam <- loadBAM(targets)
```

Overlap alignments against features:

```
> counts <- readCounts(features, bam, l=100L, maxISize=50000)
```

Perform DE/DU analysis

```
> pair <- c("CT","KD")
> group <- c(rep("CT", 4),rep("KD", 4))
> du_HNRNPC <- DUreport(counts, targets, pair, group)
```

Analyze AS using junctions:

```
> as_HNRNPC <- AsDiscover(counts=counts, targets=targets, features=features, bam=bam, l=100L, pair=pair)
```

Select TopTags (only using DU information:)

```
> binsdu <- binsDU(du_HNRNPC)
> topTagsBins <- which(binsdu$bin.fdr <= 0.1 &
                  abs(binsdu$logFC) >=0.58)
```

We can inspect visually the results of topTags. Sometimes is useful to merge BAM files by conditions. *(In this example data, control and knock down replicates were merged using `samtools merge` into* **CT.bam** *and* **KD.bam** *respectively).* Before plotting it is mandatory to install and load `Gviz` package.

```
> targetsPlot <- data.frame(bam=targets$bam,
                          sample=targets$condition,
                          color=c(rep("blue", 4),rep("red", 4)),
                          stringsAsFactors=FALSE)

> auxdf<-binsdu[topTagsBins,]
> #for simplicity, we choose one: LRR1:E005
> plotTopTags(auxdf["LRR1:E005",],
             genome,
             targetsPlot,
             output.dir="testPlots")
```

Check overlap between default DUreport (using gene expression correction and edgeR) and DUreport using DEXSeq:

```
> binsdu <- binsDU(du_HNRNPC)
> topTagsBins <- which(binsdu$bin.fdr <= 0.1 &
                       abs(binsdu$logFC) >=0.58)
```

Analyze with DEXSeq:

```
> du_DEXSeq <- DUreport_DEXSeq(counts, targets, pair, group)
> binsdx <- binsDU(du_DEXSeq)
> topTagsBinsDx <- which(binsdx$bin.fdr <= 0.1 &
                       abs(binsdx$logFC) >=0.58)
```

Plot in a Venn Diagram:

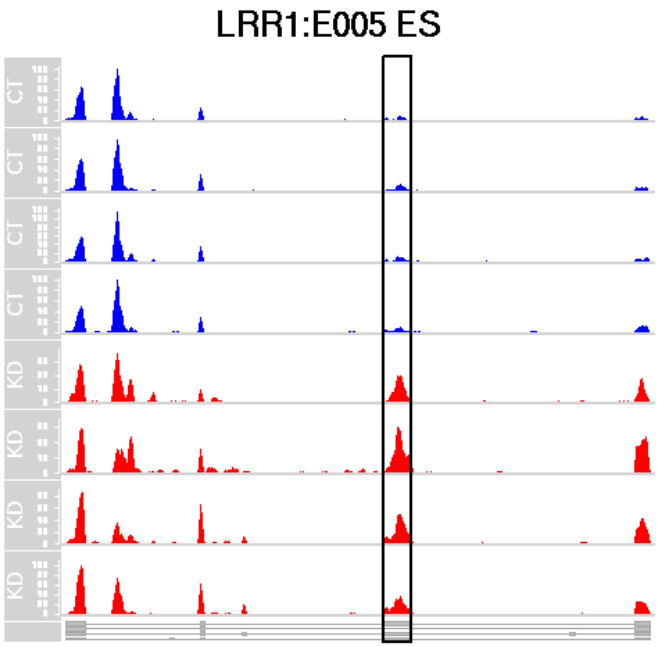


Figure 4: Plot of LRR1:E005 Exon Skipping

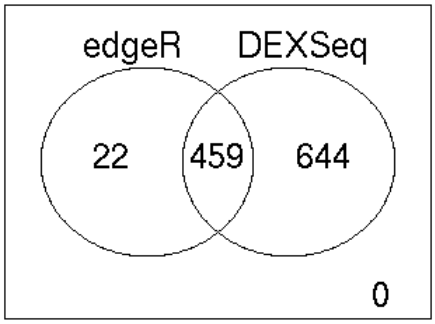


Figure 5: Top Tags bins comparison, bin.fdr less than 0.1, absolute logFC greater than 0.58

```
> all <- union(topTagsBins, topTagsBinsDx)
> auxdf <- data.frame(edgeR=all%in%topTagsBins,
                      DEXSeq=all%in%topTagsBinsDx)
> vennDiagram(auxdf)
> sessionInfo()
R version 3.3.1 (2016-06-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
```

```
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
 [1] grid      parallel  stats4    stats     graphics  grDevices utils     datasets
 [9] methods   base

other attached packages:
 [1] RNAseqData.HNRNPC.bam.chr14_0.11.0 ASpli_1.0.0
 [3] Gviz_1.18.0                        DEXSeq_1.20.0
 [5] RColorBrewer_1.1-2                 BiocParallel_1.8.0
 [7] DESeq2_1.14.0                      GenomicAlignments_1.10.0
 [9] Rsamtools_1.26.0                   Biostrings_2.42.0
[11] XVector_0.14.0                     SummarizedExperiment_1.4.0
[13] edgeR_3.16.0                       limma_3.30.0
[15] GenomicFeatures_1.26.0             AnnotationDbi_1.36.0
[17] Biobase_2.34.0                     GenomicRanges_1.26.0
[19] GenomeInfoDb_1.10.0                IRanges_2.8.0
[21] S4Vectors_0.12.0                   BiocGenerics_0.20.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.7                   locfit_1.5-9.1
 [3] biovizBase_1.22.0             lattice_0.20-34
 [5] digest_0.6.10                 mime_0.5
 [7] R6_2.2.0                      plyr_1.8.4
 [9] chron_2.3-47                  acepack_1.3-3.3
[11] RSQLite_1.0.0                 BiocInstaller_1.24.0
[13] httr_1.2.1                    ggplot2_2.1.0
[15] zlibbioc_1.20.0               data.table_1.9.6
[17] annotate_1.52.0               rpart_4.1-10
[19] Matrix_1.2-7.1                splines_3.3.1
[21] statmod_1.4.26                AnnotationHub_2.6.0
[23] geneplotter_1.52.0            stringr_1.1.0
[25] foreign_0.8-67                RCurl_1.95-4.8
[27] biomaRt_2.30.0                munsell_0.4.3
[29] shiny_0.14.1                  httpuv_1.3.3
[31] rtracklayer_1.34.0            htmltools_0.3.5
[33] nnet_7.3-12                   interactiveDisplayBase_1.12.0
[35] gridExtra_2.2.1               Hmisc_3.17-4
[37] matrixStats_0.51.0            XML_3.98-1.4
[39] bitops_1.0-6                  xtable_1.8-2
[41] gtable_0.2.0                  DBI_0.5-1
[43] magrittr_1.5                  scales_0.4.0
[45] stringi_1.1.2                 hwriter_1.3.2
[47] genefilter_1.56.0             latticeExtra_0.6-28
[49] Formula_1.2-1                 BiocStyle_2.2.0
[51] ensembldb_1.6.0               tools_3.3.1
[53] dichromat_2.0-0               BSgenome_1.42.0
[55] survival_2.39-5               colorspace_1.2-7
[57] cluster_2.0.5                 VariantAnnotation_1.20.0
```