IVAS : Identification of genetic Variants affecting Alternative Splicing

Seonggyun Han and Sangsoo Kim

May 4, 2016

Contents

1 Introduction

Alternative splicing controls relative expression ratios of mature mRNA isoforms from a single gene. Mapping studies of Splicing Quantitative Trait Loci (SQTL), a genetic variant affecting the alternative splicing, are important steps to understand gene regulations and protein activity [1]. We present an effective and user-friendly computational tool to detect SQTLs using transcript expression data from RNA-seq and genotype data, both measured on the same sample. As RNA sequencing (RNA-seq) provides insight into relatively precise measurments of expression level of transcript isoforms from a gene, it is a useful tool to analyze complicated biological phenomenons of RNA transcripts including the alternative splicing [2]. The mapping analysis uses two statistical models : Linear regression model [3] and/or Generalized linear mixed model [5].

2 The input data set

The next subsection introduces the input data. To run this tool, two experimental data sets (an expression data frame from RNA-seq and a genotype data frame) are required. Moreover, we also need a data frame for positions of SNP markers and GTF file for transcript models. As any other genome-wide analyses, it is recommended to use as many samples as possible, usually of population scale, in order to guarantee a statistically significant result.

2.1 The genotype data

The genotype data should be prepared as a simple matrix data. Each column represents an individual and its name should match that of the expression matrix described below (2.2)

	ind1	ind2	ind3	ind4
SNP1	AA	AA	AT	TT
SNP2	CG	CC	GG	CG
SNP3	TT	TT	AT	TT

2.2 The expression data

The expression matrix must comprise expresion values of transcripts from RNA-seq. We may obtain them by using alignment tools such as cufflinks. Each column represents an individual and its name should match that of the genotype matrix described above (2.1)

	ind1	ind2	ind3	ind4
transcript1	10.5	15.4	6.7	12.4
transcript2	6.4	7.2	4.5	9.2
transcript3	15.4	14.5	13.2	17.8

2.3 The SNP marker position data

To search SNPs affecting alternative splicing, a data frame comprising genomic location of each SNP is required. It consists of following columns: SNP(SNP marker name), CHR(chromosome number), and locus(SNP position).

SNP	CHR	locus
SNP1	1	4964005
SNP2	1	23513047

2.4 The transcripts model data

We need a reference GTF(General Feature Format) file including information about gene structures such as the positions of exons, introns, and transcripts of genes. The GTF file must be `TxDb` object from the *GenomicFeatures* package [4].

3 The example dataset : data from Geuvadis RNA sequencing project of 1000 Genome samples

This example uses filtered data from an origin data generated by Geuvadis RNA sequencing project, available at `http://www.geuvadis.org/web/geuvadis/RNAseq-project` [6]. The example expression data includes transcripts of 11 randomly selected genes. The genotype data comprises SNPs in those genes.

4 Loading data

For this analysis, you need to load the *IVAS* package, SNP data, expression data, SNP position data, and `TxDb` object from GTF.
Loading *IVAS* package :

```
> library(IVAS)
```

Loading expression data :

```
> data(sampleexp)
```

Loading SNP data :

```
> data(samplesnp)
```

Loading SNP position data :

```
> data(samplesnplocus)
```

Loading `TxDb` object :

```
> sampleDB <- system.file("extdata", "sampleDB", package="IVAS")
> sample.Txdb <- loadDb(sampleDB)
```

If you want to create the `TxDb` object from a GTF file, you need to use the `makeTxDbFromGFF` function in the *GenomicFeatures* package.

5 Running the IVAS package for detecting SQTLs

5.1 Separating the `TxDb` object based on a single chromosome.

The `chrseparate` function separates the `TxDb` object based on a single chromosome for mapping analysis of expression and genotype in a single chromosome.
To use this function :

```
> filtered.txdb <- chrseparate(sample.Txdb,19)
> filtered.txdb
```

```
TxDb object:
# Db type: TxDb
# Supporting package: GenomicFeatures
# Data source: /data/sampleGTF
# Organism: Homo Sapience
# miRBase build ID: NA
# transcript_nrow: 64
# exon_nrow: 239
```

```
# cds_nrow: 184
# Db created by: GenomicFeatures package from Bioconductor
# Creation time: 2015-03-11 03:54:50 +0900 (Wed, 11 Mar 2015)
# GenomicFeatures version at creation time: 1.19.6
# RSQLite version at creation time: 1.0.0
# DBSCHEMAVERSION: 1.0
```

In this example, We filter the TxDb object with only the chromosome 6.

5.2 Finding alternative exons of a single gene

The `findAlternative` function finds flanking introns of alternative exons from a single gene. To run this function, it requires trans.exon.range, trans.intron.range, and txTable. The trans.exon.range is a range of exons based on transcripts using the exonsBy function in *GenomicFeatures* package. The trans.intron.range is a range of introns based on transcripts using the `intronsBy` function in *GenomicFeatures* package. The txTable has information about trancripts(start site, chromosome number, etc).

```
> trans.exon.range <- exonsBy(filtered.txdb,by="tx")
> trans.intron.range <- intronsByTranscript(filtered.txdb)
> txTable <- select(filtered.txdb, keys=names(trans.exon.range),
+                   columns=c("TXID","TXNAME","GENEID","TXSTART","TXEND"),keytype="TXID")
> Altvalue <- findAlternative("ENSG00000170889",txTable,trans.exon.range,
+                             trans.intron.range,19)
> Altvalue
```

```
$alterIntron
GRanges object with 6 ranges and 0 metadata columns:
      seqnames              ranges strand
         <Rle>           <IRanges>  <Rle>
  [1]       19 [54704757, 54705027]      +
  [2]       19 [54705478, 54710143]      +
  [3]       19 [54710331, 54711265]      +
  [4]       19 [54704830, 54705027]      +
  [5]       19 [54710331, 54710420]      +
  [6]       19 [54705478, 54711265]      +
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths

$tableBygene
  TXID          GENEID          TXNAME  TXSTART    TXEND
1   40 ENSG00000170889 ENST00000302907 54704610 54711515
2   41 ENSG00000170889 ENST00000391752 54704740 54711515
3   42 ENSG00000170889 ENST00000402367 54704740 54711515
4   43 ENSG00000170889 ENST00000391751 54704742 54711515
5   44 ENSG00000170889 ENST00000391753 54705001 54711515
6   45 ENSG00000170889 ENST00000441429 54705028 54711498

$exonRange
GRangesList object of length 6:
$40
GRanges object with 5 ranges and 3 metadata columns:
      seqnames              ranges strand |    exon_id   exon_name exon_rank
```

```
         <Rle>            <IRanges>  <Rle> | <integer> <character> <integer>
  [1]       19 [54704610, 54704756]      + |       172        <NA>         1
  [2]       19 [54705028, 54705149]      + |       176        <NA>         2
  [3]       19 [54705355, 54705477]      + |       177        <NA>         3
  [4]       19 [54710144, 54710330]      + |       178        <NA>         4
  [5]       19 [54711266, 54711515]      + |       181        <NA>         5

$41
GRanges object with 5 ranges and 3 metadata columns:
      seqnames              ranges strand | exon_id exon_name exon_rank
  [1]       19 [54704740, 54704829]      + |     173      <NA>         1
  [2]       19 [54705028, 54705149]      + |     176      <NA>         2
  [3]       19 [54705355, 54705477]      + |     177      <NA>         3
  [4]       19 [54710144, 54710330]      + |     178      <NA>         4
  [5]       19 [54711266, 54711515]      + |     181      <NA>         5

$42
GRanges object with 5 ranges and 3 metadata columns:
      seqnames              ranges strand | exon_id exon_name exon_rank
  [1]       19 [54704740, 54704829]      + |     173      <NA>         1
  [2]       19 [54705028, 54705149]      + |     176      <NA>         2
  [3]       19 [54705355, 54705477]      + |     177      <NA>         3
  [4]       19 [54710144, 54710330]      + |     178      <NA>         4
  [5]       19 [54710421, 54711515]      + |     180      <NA>         5

...
<3 more elements>
-------
seqinfo: 1 sequence from an unspecified genome; no seqlengths

$intronRange
GRangesList object of length 6:
$40
GRanges object with 4 ranges and 0 metadata columns:
      seqnames              ranges strand
         <Rle>            <IRanges>  <Rle>
  [1]       19 [54704757, 54705027]      +
  [2]       19 [54705150, 54705354]      +
  [3]       19 [54705478, 54710143]      +
  [4]       19 [54710331, 54711265]      +

$41
GRanges object with 4 ranges and 0 metadata columns:
      seqnames              ranges strand
  [1]       19 [54704830, 54705027]      +
  [2]       19 [54705150, 54705354]      +
  [3]       19 [54705478, 54710143]      +
  [4]       19 [54710331, 54711265]      +

$42
GRanges object with 4 ranges and 0 metadata columns:
```

```
      seqnames               ranges strand
  [1]       19 [54704830, 54705027]      +
  [2]       19 [54705150, 54705354]      +
  [3]       19 [54705478, 54710143]      +
  [4]       19 [54710331, 54710420]      +

...
<3 more elements>
-------
seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

In this example, we will search SQTLs in the ENSG00000170889.

5.3 Finding SNPs which belongs to alternative exons and flanking introns

The `overlapsnp` function searches SNPs in alternative exons and the flanking introns.

```
> ch.snp.locus <- as.matrix(samplesnplocus[samplesnplocus[,2] == 19,])
> ch.snps <- matrix(ch.snp.locus[is.element(ch.snp.locus[,1],rownames(samplesnp)),],
+                   ncol=3,byrow=FALSE)
> ch.snps.range <- GRanges(seqnames=Rle(19),ranges=IRanges(start=as.integer(ch.snps[,3]),
+                          end=as.integer(ch.snps[,3])),metadata=ch.snps[,1])
> overlapsnp <- findOversnp(Altvalue,ch.snps.range)
> overlapsnp
```

```
  range                snp
1 "54704757-54705027" "rs3810232"
```

5.4 Finding SQTLs

Using the output data from the `Altvalue` function and the `overlapsnp` function, significant SNPs affecting the alternative splicing can be identified by the sqtlfinder function.

```
> sqtl.result <- sqtlfinder(Altvalue,overlapsnp,sampleexp,samplesnp,"lm")
> sqtl.result
```

```
  SNP          CHR  targetExon           intron of SNP        type   P.value
1 "rs3810232" "19" "54704610-54704756" "54704757-54705027" "A5SS" "9.60009815744455e-14"
  per.P.value gene               method strand
1 "sig"       "ENSG00000170889" "lm"   "+"
```

In this example, We will run the function with the linear regression model.

6 Identification of SQTLs in multiple genes

The functions in the *IVAS* package perform mapping analysis using a single gene. However, the `MsqtlFinder` function in the *IVAS* package enables one to analyze multiple genes using the multi-thread version of `foreach` function. Moreover, it joins the results on the genes from sqtlfinder function and calculates the FDR using P-values of the results.

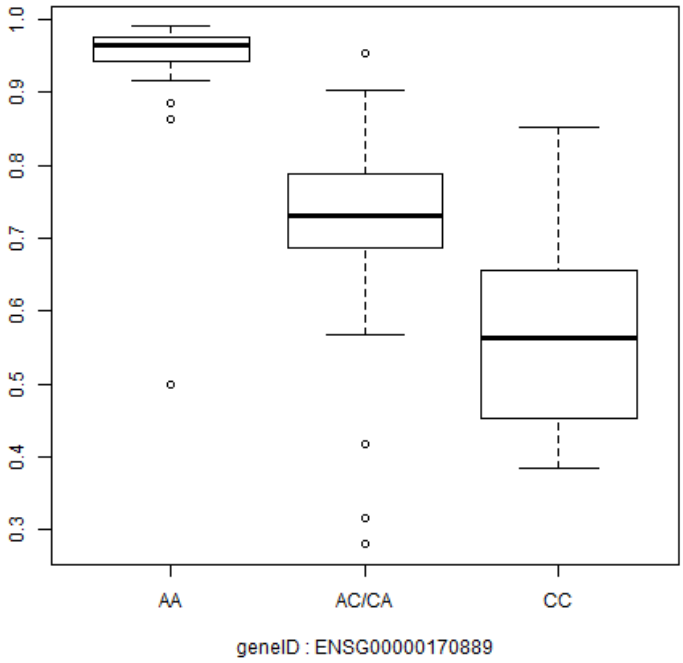
```
> final.result <- MsqtlFinder(sampleexp,samplesnp,samplesnplocus,sample.Txdb,"lm",1)
```

`MsqtlFinder` shows chromosome numbers during mapping analysis. The last argument denotes the number of threads(a single processor in this example)

7 Visualizing the result

To visualize the results into boxplot, the *IVAS* package provides the saveBplot function. Using the dataframe from the output of `sqtlfinder` or `MsqtlFinder` function, we can make the boxplot.

```
> saveBplot(sqtl.result,sampleexp,samplesnp,samplesnplocus,filtered.txdb,"./result")
```



The output png files are saved in "result" folder.

8 Session Information

```
R version 3.3.0 RC (2016-04-25 r70549)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server 2008 R2 x64 (build 7601) Service Pack 1

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

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

```
other attached packages:
[1] IVAS_1.4.0              GenomicFeatures_1.24.0 AnnotationDbi_1.34.0
[4] Biobase_2.32.0          GenomicRanges_1.24.0   GenomeInfoDb_1.8.0
[7] IRanges_2.6.0           S4Vectors_0.10.0       BiocGenerics_0.18.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.4.5               XVector_0.12.0              MASS_7.3-45
 [4] splines_3.3.0               zlibbioc_1.18.0             GenomicAlignments_1.8.0
 [7] BiocParallel_1.6.0          doParallel_1.0.10           lattice_0.20-33
[10] foreach_1.4.3               minqa_1.2.4                 tools_3.3.0
[13] grid_3.3.0                  SummarizedExperiment_1.2.0 nlme_3.1-127
[16] DBI_0.4                     iterators_1.0.8             lme4_1.1-12
[19] Matrix_1.2-6                nloptr_1.0.4                rtracklayer_1.32.0
[22] codetools_0.2-14            bitops_1.0-6                RCurl_1.95-4.8
[25] biomaRt_2.28.0              RSQLite_1.0.0               Biostrings_2.40.0
[28] Rsamtools_1.24.0            XML_3.98-1.4                BiocStyle_2.0.0
```

References

[1] Keyan Zhao, Zhi-xiang Lu, Juw Won Park, Qing Zhou, Yi Xing. 2013. GLiMMPS: robust statistical model for regulatory variation of alternative splicing using RNA-seq data. Genome Biol 14, R74.

[2] Joseph K. Pickrell, John C. Marioni, Athma A. Pai, Jacob F. Degner, Barbara E. Engelhardt, Everlyne Nkadori, Jean-Baptiste Veyrieras, Matthew Stephens, Yoav Gilad, Jonathan K. Pritchard. 2010. Understanding mechanisms underlying human gene expression variation with RNA sequencing. Nature 464, 768-722.

[3] N.E. Breslow and D.G. Clayton. 1993. Approximate Inference in Generalized Linear Mixed Models. Journal of the American Statistical Association 88 421: 9-25.

[4] Michael Lawrence, et al. 2013. Software for Computing and Annotating Genomic Ranges. PLoS Comput Biol. 9(8): e1003118.

[5] Chambers, J. M. 1992. Linear models. Chapter 4 of Statistical Models in S eds J. M. Chambers and T. J. Hastie, Wadsworth, and Brooks Cole.

[6] Tuuli Lappalainen, et al. 2013. Transcriptome and genome sequencing uncovers functional variation in humans. Nature 501, 506-511.