SplicingCompass Tutorial

Moritz Aschoff

February 25, 2013

Abstract

SplicingCompass predicts genes that are differentially spliced between two different conditions using RNA-seq data. SplicingCompass is an open source package developed in R[R D10] and licensed under the GNU General Public License.

1 Mapping of RNA-seq Reads and Counting

In order to use SplicingCompass two things have to be done beforehand:

1. Use your favorite read mapping algorithm and produce bam files for every sample. We tested SplicingCompass with bam files produced with TopHat v1.2.0[TPS09].
2. Use the coverageBed command from BEDTools[QH10] (or another equivalent method) to count the number of reads that align to all exons in annotated union transcripts (see listing 1). We provide a processed annotation of union transcripts that is based on the UCSC CCDS annotation[PHH+06] (file *ccdsMergedTranscripts.gtf*). R-scripts to build your own annotation can be provided on request.

Listing 1: Counting reads with coverageBed

```
1 coverageBed -split -abam sample_N.bam -b ccdsMergedTranscripts.gtf >
2 covBedCountedHits_sample_N.gff
```

2 Installing and Running SplicingCompass

SplicingCompass can be installed from the command line via:

Listing 2: Installing the SplicingCompass package

```
3 R CMD INSTALL -l /path/to/library SplicingCompass_1.0.tar.gz
```

The part "-l /path/to/library" can be omitted if the standard library path should be used (which might require corresponding user privileges).
(cmp. `http://cran.r-project.org/doc/manuals/R-admin.html#Installing-R-under-Unix_002dalikes`).

After starting the R process, three steps are necessary to run SplicingCompass:

1. Constructing an object of class *ExperimentInfo* by setting some basic information about the experiment and the count files produced by coverageBed and the mapping algorithm (see sections 1 and 2.1).
2. Constructing an object of class *CountTable* that holds processed and normalised read counts (see section 2.2).
3. Constructing an object of class *SplicingCompass* that calculates geometric angles and tests for statistical significance (see section 2.3).

2.1 Providing experimental information

First, the library is loaded and a package description is shown (see listing 3, lines 1-2). Again, the lib.loc parameter can be omitted if the standard library path was used during installation (the path that is shown by .libPaths()[1] in R). Second, an ExperimentInfo object is created and a description for the experiment is set (lines 4-5). Third, a description for the two groups of interest is set and the number of samples associated with each group is defined (lines 7-9). The groupName parameters are character strings and must not contain whitespaces or special characters. The sampleNumsGroup parameters are numeric vectors that indicate which samples (i.e. which files provided via "setCovBedCountFiles" in lines 11-19 and "setJunctionBedFiles" in lines 21-29) belong to which group. The file paths that contain read counts (lines 11-19) and junction reads (lines 21-29) must be provided (cmp. section 1). The order of the files must be consistent with the information provided in "setGroupInfo" (lines 7-9). The junction reads must be provided in bed format (e.g. as reported by TopHat). Finally, the reference annotation that was used for counting must be provided (lines 31-33). This is necessary to resolve which exons are connected by junction reads. The reference annotation format can be adjusted via setReferenceAnnoationFormat (lines 32-33). E.g. in "ccdsMergedTranscripts.gtf", the attribute field has the the format "geneSymbol "value";" A simple consistency check is done in line 35 and must return TRUE.

Listing 3: Example for setting up an ExperimentInfo object

```
1  library("SplicingCompass",lib.loc="/path/to/library")
2  packageDescription("SplicingCompass")
3
4  expInf=new("ExperimentInfo")
5  expInf=setDescription(expInf,"ControlVsCancer")
6
7  expInf=setGroupInfo(expInf,
8  groupName1="control",sampleNumsGroup1=1:3,
9  groupName2="cancer",sampleNumsGroup2=4:6)
10
11 covBedCountFilesControl=c(
12 "/path/to/covBedCountedHits_sample_1.gff",
13 "/path/to/covBedCountedHits_sample_2.gff",
14 "/path/to/covBedCountedHits_sample_3.gff")
15 covBedCountFilesCancer=c(
16 "/path/to/covBedCountedHits_sample_4.gff",
17 "/path/to/covBedCountedHits_sample_5.gff",
18 "/path/to/covBedCountedHits_sample_6.gff")
19 expInf=setCovBedCountFiles(expInf,c(covBedCountFilesControl,covBedCountFilesCancer))
20
21 junctionBedFilesControl=c(
22 "/path/to/junctionsBed_sample_1.bed",
23 "/path/to/junctionsBed_sample_2.bed",
24 "/path/to/junctionsBed_sample_3.bed")
25 junctionBedFilesCancer=c(
26 "/path/to/junctionsBed_sample_4.bed",
27 "/path/to/junctionsBed_sample_5.bed",
28 "/path/to/junctionsBed_sample_6.bed")
29 expInf=setJunctionBedFiles(expInf,c(junctionBedFilesControl,junctionBedFilesCancer))
30
31 expInf=setReferenceAnnotation(expInf,"ccdsMergedTranscripts.gtf")
32 referenceAnnotationFormat=list(IDFieldName="geneSymbol",idValSep="␣")
33 expInf=setReferenceAnnotationFormat(expInf,referenceAnnotationFormat)
```

```
34
35 checkExperimentInfo(expInf)
```

2.2 Constructing an object of class CountTable

Once the ExperimentInfo object is defined a corresponding CountTable object can be constructed:

Listing 4: Constructing an object of class CountTable

```
36 countTable=new("CountTable")
37 countTable=setExperimentInfo(countTable,expInf)
38 countTable=constructCountTable(countTable,nCores=1,printDotPerGene=TRUE)
```

The number of cpu cores can be chosen for computation. If $nCores > 1$, the "multicore" package has to be loaded first via "library("multicore")". By default, dots are printed during computation. For demonstration purposes, an already constructed CountTable object comparing brain vs liver samples on chromosome 8 can be loaded into R:

Listing 5: loading an example CountTable object

```
39 countTable=get(load("countTable_BrainLiver_chr8.save"))
```

2.3 Constructing an object of class SplicingCompass

Finally, a SplicingCompass object can be constructed from the CountTable object. The minimal number of junction reads (summed over all samples) that is required to support a junction can be chosen (defaults to 5). In addition, the number of cpu cores can be chosen for computation. If $nCores > 1$, the "multicore" package has to be loaded first via "library("multicore")".

Listing 6: Constructing an object of class SplicingCompass

```
40 sc=new("SplicingCompass")
41 sc=constructSplicingCompass(sc,countTable,minOverallJunctionReadSupport=5,nCores=1)
```

Significant genes and p-values can be obtained from this object (see listing 7) and gene profiles can be plotted with the method "plotGeneProfile" (see listing 9). "initSigGenesFromResults" (line 42) selects significant genes according to a given significance level. Significant gene symbols can then be obtained via "getSignificantGeneSymbols" (line 43). The parameter "adjusted" defines whether to use p-values adjusted for multiple testing by the Benjamini and Hochberg procedure or not (defaults to TRUE).

Listing 7: Obtaining significant genes

```
42 sc=initSigGenesFromResults(sc,adjusted=TRUE,threshold=0.1)
43 sigGenes=getSignificantGeneSymbols(sc)
```

A data frame that contains all tested genes and corresponding p-values can be obtained via "getResultTable":

Listing 8: Obtaining a data frame that contains all tested genes and corresponding p-values

```
44 resTab=getResultTable(sc)
```

"plotGeneProfile" plots a read count profile for an arbitrary gene represented in the CountTable object. The read count profile is a normalised representation of the read coverage for all exons and exon-exon junctions from the union transcript of that gene (see figure 1). The normalised profile of a gene can be plotted by directly providing a symbol (line 45). In line 46 the first symbol is used that was obtained from "getSignificantGeneSymbols". The minimal number of junction reads (summed over all samples) required to support a junction can be chosen (defaults to 5).

Listing 9: Plotting gene profiles

```
plotGeneProfile(countTable,"SLC7A2",minOverallJunctionReadSupport=5)
plotGeneProfile(countTable,sigGenes[1],minOverallJunctionReadSupport=5)
```

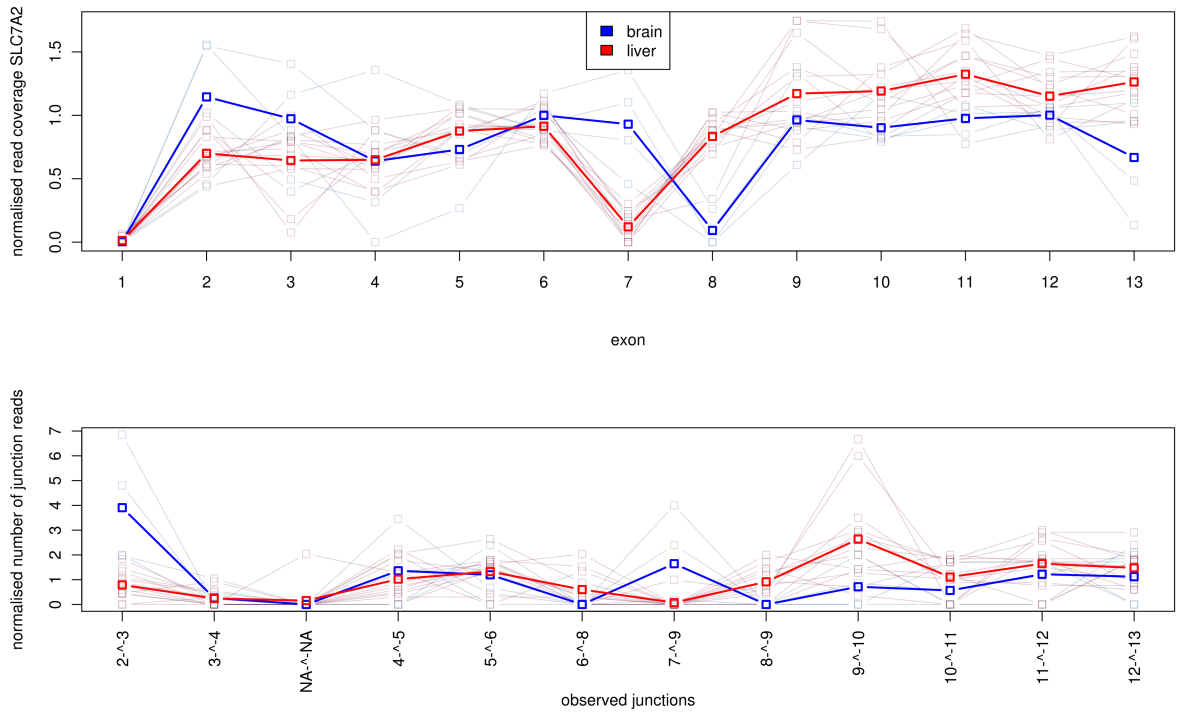

Figure 1: result of plotGeneProfile

References

[PHH+06] Kim D Pruitt, Jennifer Harrow, Rachel A Harte, Craig Wallin, Mark Diekhans, Donna R Maglott, Steve Searle, Catherine M Farrell, Jane E Loveland, Barbara J Ruef, Elizabeth Hart, Marie-marthe Suner, Melissa J Landrum, Bronwen Aken, Sarah Ayling, Robert Baertsch, Julio Fernandez-banet, Joshua L Cherry, Val Curwen, Michael Dicuccio, Manolis Kellis, Jennifer Lee, Michael F Lin, Michael Schuster, Andrew Shkeda, Clara Amid, Garth Brown, Oksana Dukhanina, Adam Frankish, Jennifer Hart, Bonnie L Maidak, Jonathan Mudge, Michael R Murphy, Terence Murphy, Jeena Rajan, Bhanu Rajput, Lillian D Riddick, Catherine Snow, Charles Steward, David Webb, Janet A Weber, Laurens Wilming, Wenyu Wu, Ewan Birney, David Haussler, Tim Hubbard, James Ostell, Richard Durbin, and David Lipman. The consensus coding sequence (CCDS) project: Identifying a common protein-coding gene set for the human and mouse genomes. Genome Research, pages 1316–1323, 2006.

[QH10] Aaron R Quinlan and Ira M Hall. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics (Oxford, England), 26(6):841–2, March 2010.

[R D10] R Development Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, 2010.

[TPS09] Cole Trapnell, Lior Pachter, and Steven L Salzberg. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics (Oxford, England), 25(9):1105–11, 2009.