AllelicImbalance

Jesper Robert Gådin and Lasse Folkersen

October 14, 2013

1 AllelicImbalance

This *AllelicImbalance* package contains functions for investigating allelic imbalance effects in RNA-seq data. Maternal and paternal alleles could be expected to show identical transcription rate, resulting in a 50%-50% mix of maternal and paternal mRNA in a sample. However, this turns out to sometimes not be the case. The most extreme example is the X-chromosome inactivation in females, but many autosomal genes also have deviations from identical transcription rate. The causes of this are not always known, but one likely cause is the difference in DNA, namely heterozygous SNPs, affecting enhancers, promoter regions, splicing and stability. Identifying this allelic imbalance is therefore of interest to the characterization of the genome and the aim of the *AllelicImbalance* package is to facilitate this.

2 Simple example of building an ASEset object

In this section we will walk through the various ways an `ASEset` object can be created. The ASEset object has the SummarizedExperiment as parent class, and all functions you can apply on this class you can also apply on an ASEset. Although the preprocessing of RNA-seq data is not the primary focus of this package, it is a necessary step before analysis. There exists several different methods for obtaining a bam file, and this section should just be considered an example. For further details we refer to the web-pages of tophat, bowtie, bwa and samtools found in the links section at the end of this document.

```
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR009/ERR009135/*
bowtie -q --best --threads 5 --sam hg19 +
>      -1 ERR009135_1.fastq.gz -2 ERR009135_2.fastq.gz "ERR009135.sam"
samtools view -S -b ERR009135.sam > ERR009135.bam
```

In the above code one paired-end RNA sequencing sample is downloaded and aligned to the human genome, then converted to bam using samtools. The resulting bam files can be the direct input to the AllelicImbalance package. Other aligners can be used as well, as long as bam files are provided as input. The example code following illustrates how to use the import mechanism on a chromosome 17-located subset of 20 RNA-seq experiments of HapMap samples. The output is an `ASEset` object containing allele counts for all heterozygote coding SNPs in the region.

```
> searchArea <- GRanges(seqnames = c("17"),ranges = IRanges(79478301,79478361))
> pathToFiles <- system.file("extdata/ERP000101_subset", package="AllelicImbalance")
> reads <- impBamGAL(pathToFiles,searchArea,verbose=FALSE)
> heterozygotePositions <- scanForHeterozygotes(reads,verbose=FALSE)
> countList <- getAlleleCount(reads, heterozygotePositions, verbose=FALSE)
> a.simple <- ASEsetFromCountList(heterozygotePositions,countList)
> a.simple
```

```
class: ASEset
dim: 3 20
exptData(0):
assays(2): countsNonStranded mapBias
rownames(3): chr17_79478331 chr17_79478334 chr17_79478287
rowData metadata column names(0):
colnames(20): ERR009097.bam ERR009102.bam ... ERR009160.bam
  ERR009167.bam
colData names(0):
```

3 Building an ASEset object using Bcf files

If more than a few genes and a few samples are analyzed we recommend that a SNP-call is instead made using the samtools mpileup function (see links section). The `scanForHeterozygotes` function is merely a simple SNP-caller and it is not as computationally optimized as mpileup. In this bash code we download reference sequence for chromosome 17 and show how to generate mpileup calls on one of the HapMap samples that were described above.

```
wget ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/chr17.fa.gz
samtools mpileup -uf hg19.fa ERR009135.bam | bcftools view -bvcg - > ERR009135.bcf
```

Samtools mpileup generates by default a Vcf file which contains SNP and short INDEL positions. Piping the output to bcftools we get its binary equivalent (Bcf), which takes less space and can be queried more effective. With the Bcf files the process of generating an ASEset object starts with a call to the `impBcfGR` function instead. This function will import the Bcf file containing all SNP calls that were generated with the samtools mpileup function.

```
> BcfGR <- impBcfGR(pathToFiles,searchArea,verbose=FALSE)
> countListBcf <- getAlleleCount(reads, BcfGR,verbose=FALSE)
> a.bcf <- ASEsetFromCountList(BcfGR, countListBcf)
```

4 Using strand information

Many RNA-seq experiments do not yield useful information on the strand from which a given read was made. This is because they involve a step in which a double-stranded cDNA is created without tracking strand-information. Some RNA-seq setups do however give this information and in those cases it is important to keep track of strand in the ASE-experiment. The example data from above is from an experiment which created double-stranded cDNA before labelling and so the '+' and '-' information in it is arbitrary. However, if we

assume that the information has strand information, then the correct procedure is as follows:

```
> plus <- getAlleleCount(reads, heterozygotePositions, strand="+",verbose=F)
> minus <- getAlleleCount(reads, heterozygotePositions, strand="-",verbose=F)
> a.stranded <-
+ ASEsetFromCountList(
+ heterozygotePositions,
+ countListPlus=plus,
+ countListMinus=minus
+ )
> a.stranded
```

```
class: ASEset
dim: 3 20
exptData(0):
assays(3): countsPlus countsMinus mapBias
rownames(3): chr17_79478331 chr17_79478334 chr17_79478287
rowData metadata column names(0):
colnames(20): ERR009097.bam ERR009102.bam ... ERR009160.bam
  ERR009167.bam
colData names(0):
```

The main effect of doing this, is in the plotting functions which will separate reads from different strands if they are specified as done here. It is important, however, to make sure that the imported RNA-seq experiment does in fact have proper labeling and tracking of strand information before proceeding with this method.

5 Two useful helper functions

At this stage it is worth highlighting two useful helper functions that both uses existing BioC annotation objects. One is the `getAreaFromGeneNames` which quickly retrieves the above mentioned `searchArea` when given just genesymbols as input. The other other is the `getSnpIdFromLocation` function which attempts to rename location-based SNP names to established rs-IDs in case they exist. These functions work as follows:

```
> #Getting searchArea from genesymbol
> library(org.Hs.eg.db )
> searchArea<-getAreaFromGeneNames("ACTG1",org.Hs.eg.db)
```

```
Found all requested genes in annotation
```

```
> #Getting rs-IDs
> library(SNPlocs.Hsapiens.dbSNP.20120608)
> updatedGRanges<-getSnpIdFromLocation(rowData(a.simple), SNPlocs.Hsapiens.dbSNP.20120608)
```

```
Replacing position-based SNP name with rs-ID for 1 SNP(s)
```

```
> rowData(a.simple)<-updatedGRanges
>
```

6 Adding phenotype data

Typically an RNA-seq experiment will include additional information about each sample. It is an advantage to include this information when creating an ASEset because it can be used for subsequent highlights or subsetting in plotting and analysis functions.

```
> #simulate phenotype data
> pdata <- DataFrame(
+         Treatment=sample(c("ChIP", "Input"),length(reads),replace=TRUE),
+         Gender=sample(c("male", "female"),length(reads),replace=TRUE),
+         row.names=paste("individual",1:length(reads),sep=""))
> #make new ASEset with pdata
> a.new <- ASEsetFromCountList(
+                 heterozygotePositions,
+                 countList,
+                 colData=pdata)
> #add to existing object
> colData(a.simple) <- pdata
>
```

7 Statistical analysis of an ASEset object

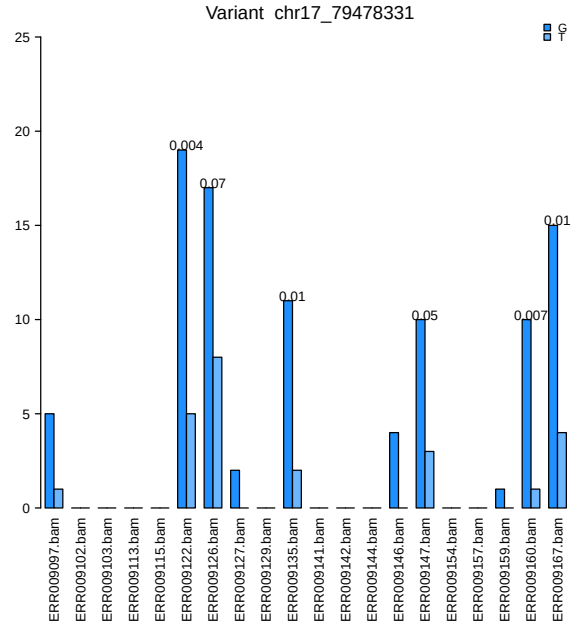
One of the simplest statistical test for use in allelic imbalance analysis is the chi-square test. This test assumes that the uncertainty of ASE is represented by a normal distribution around an expected mean (i.e 0.5 for equal expression). A significant result suggests an ASE event. Every strand is tested independently.

```
> #use a subset for tests
> a2 <- a.stranded[,5:10]
> #two types of tests
> binom.test(a2,"+")

     chr17_79478331 chr17_79478334 chr17_79478287
[1,]             NA             NA             NA
[2,]    0.006610751      0.2668457      0.7265625
[3,]    0.107752144             NA             NA
[4,]             NA             NA             NA
[5,]             NA             NA             NA
[6,]    0.022460938      0.0703125             NA

> chisq.test(a2,"-")

     chr17_79478331 chr17_79478334 chr17_79478287
[1,]             NA             NA             NA
[2,]             NA             NA      0.7962534
[3,]             NA             NA             NA
[4,]             NA             NA             NA
[5,]             NA             NA             NA
[6,]             NA             NA             NA
```

8 Plotting of an ASEset object

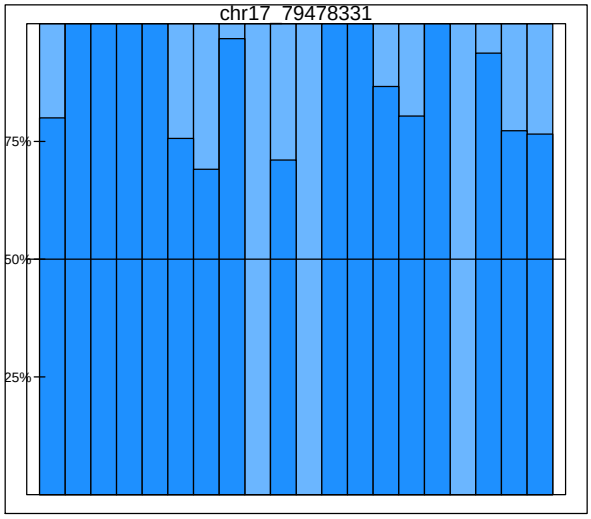
The `barplot` function for `ASEset` objects plots the read count of each allele in each sample. This is useful for getting a very detailed view of individual SNPs in few samples. As can be seen below, four samples from the HapMap data contains a strong imbalance at the chr17:79478331 position on the plus strand. By default the p-value is calculated by a chi-square test. To use other test results the arguments `testValue` and `testValue2` can be used. When the counts for one allele are below 5 for one allele the chi-square test returns NA. This is why there is no P-value above the first bar in the example below.

```
> barplot(a.stranded[1],strand="+")
> #use other test
> btp <- binom.test(a.stranded[1],"+")
> barplot(a.stranded[1],strand="+", testValue=btp)
>
```



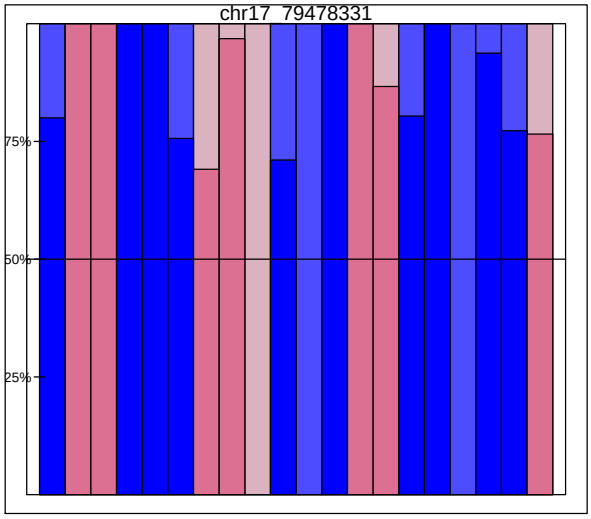
Another example of plotting that is useful is the one invoked with the plotting `type` argument "fraction". This plotting mechanism is useful to illustrate more SNPs and more samples in less space than the standard plot. As can be seen here several other samples are not heterozygote at the chr17:79478331 location.

```
> barplot(a.simple,type="fraction")
```



A typical question would be to ask why certain heterozygote samples have allele specific expression. The argument `sampleColour` argument allows for different highligts such as illustrated here below for gender. This could also be used to highlight based on genotype of proximal non-coding SNPs if available.

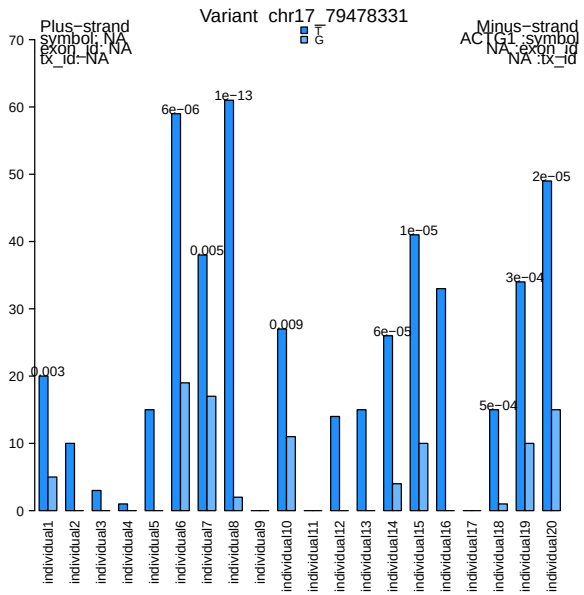
```
> sampleColour<-rep("palevioletred",ncol(a.simple))
> sampleColour[colData(a.simple)[,"Gender"]%in%"male"] <- "blue"
> barplot(a.simple[1],type="fraction",sampleColour=sampleColour)
>
```



9 Plot with annotation

It is often of interest to combine the RNA sequencing data with genomic annotation information from online databases. For this purpose there is a function to extract variant specific annotation such as gene, exon, transcript and CDS.

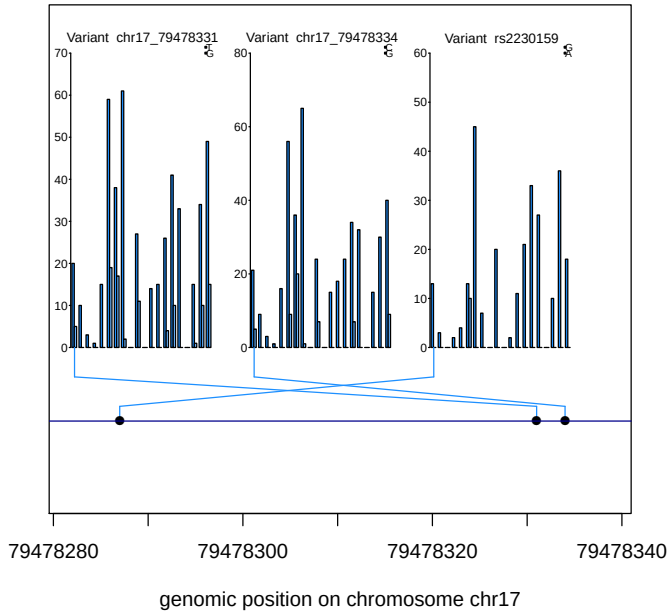
```
> library(org.Hs.eg.db)
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> barplot(a.simple[1],OrgDb=org.Hs.eg.db,TxDb=TxDb.Hsapiens.UCSC.hg19.knownGene)
>
```



10 locationplot

Finally a given gene or set of proximal genes will often have several SNPs close to each other. It is of interest to investigate all of them together, in connection with annotation information. This can be done using the `locationplot` function. This function in its simplest form just plot all the SNPs in an ASEset distributed by genomic location. Additionally it contains methods for including gene-map information through the arguments `OrgDb` and `TxDb`.

```
> #using count type
> locationplot(a.simple,type="count")
> #use annotation
> locationplot(a.simple,OrgDb=org.Hs.eg.db,TxDb=TxDb.Hsapiens.UCSC.hg19.knownGene)
>
```



11 Conclusion

In conclusion we hope that you will find this package useful in the investigation of the genetics of RNA-seq experiments. The various import functions should assist in the task of actually retrieving allele counts for specific nucleotide positions from all RNA-seq reads, including the non-trivial cases of intron-spanning reads. Likewise, the statistical analysis and plotting functions should be helpful in discovering any allele specific expression patterns that might be found in your data.

12 Links

Bowtie `http://bowtie-bio.sourceforge.net`
BWA `http://bio-bwa.sourceforge.net/`
Samtools `http://samtools.sourceforge.net/`
Samtools pileup `http://samtools.sourceforge.net/mpileup.shtml`

Session Info

```
> sessionInfo()

R version 3.0.2 (2013-09-25)
Platform: x86_64-unknown-linux-gnu (64-bit)
```

```
locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] TxDb.Hsapiens.UCSC.hg19.knownGene_2.10.1
 [2] GenomicFeatures_1.14.0
 [3] SNPlocs.Hsapiens.dbSNP.20120608_0.99.9
 [4] BSgenome_1.30.0
 [5] Biostrings_2.30.0
 [6] org.Hs.eg.db_2.10.1
 [7] RSQLite_0.11.4
 [8] DBI_0.2-7
 [9] AnnotationDbi_1.24.0
[10] Biobase_2.22.0
[11] AllelicImbalance_1.0.0
[12] GenomicRanges_1.14.0
[13] XVector_0.2.0
[14] IRanges_1.20.0
[15] BiocGenerics_0.8.0

loaded via a namespace (and not attached):
[1] RCurl_1.95-4.1     Rsamtools_1.14.0   XML_3.98-1.1       biomaRt_2.18.0
[5] bitops_1.0-6       rtracklayer_1.22.0 stats4_3.0.2       tools_3.0.2
[9] zlibbioc_1.8.0

>
```