Nicholas Hebert

Using R to Examine Allele Specific Expression

Background:

Allele specific expression (ASE) is the process through which one of the parent's alleles is preferentially expressed in a diploid or polyploid offspring which can be due to both epigenetic and genetic factors. Normally, diploid organisms have two alleles for each gene, both of which are equally expressed. However, in the case of ASE, one of those alleles is expressed more so than the other. Some cases of ASE can be relatively mild where one allele is only expressed slightly more than the other, and some cases can be more severe. Regardless of the intensity of the event, ASE is an incredibly important process across multiple organisms. An example within the human genome is the ASE of the death associated protein kinase 1 (DAPK1) that contributes to developing late onset Alzheimer's disease (LOAD) (Li et al., 2006). Another example within the human genome is the ASE of the adenomatous polyposis coli (APC) gene which is a contributing factor to developing familial adenomatous polyposis. Left untreated, this can develop into colorectal cancer (Curia 2012). There are many more diseases and conditions which ASE is thought to be associated with. As such, there is a great deal of importance in being able to identify one of the causes behind those diseases and conditions. Currently, there are multiple ASE analysis packages available which function to detect ASE events in heterozygous sites.

Bioinformatic tool used:

The bioinformatic tool used through this walkthrough is the “AllelicImbalance” package that is loaded into R (Gadin et al., 2015). There exist other platforms which can have ASE packages loaded, but R will be the platform used throughout this tutorial. The AllelicImbalance package analyzes ASE via a chi square and/or binomial test on an ASESetListObject. This object is created based on allele counts from heterozygous sites within a specified range of a dataset. The statistical tests are operating under some assumptions: there is equal expression between each of the alleles for a gene and each gene is tested independently. As such, the null hypothesis being tested would be that the alleles for a gene analyzed are both equally expressed. In contrast, the alternative hypothesis would be that the alleles are not equally expressed. As mentioned previously, there are other packages that can analyze ASE within R, and there are a lot of similarities between those packages and the AllelicImbalance package. For instance, the other packages compared, MBASED (Mayba, 2020) and ASEP (Jiaxin et al., 2019), both have the same assumptions for the statistical tests. In addition to this, the packages have similar data requirements in that they all build an object based off of a RangedSummarizedExperiment. There are many other similarities, but there are also some notable differences between the packages. One of the differences would be that the AllelicImbalance package allows you to add in phenotypic data. Another notable difference would be that you can obtain a search area for a gene based on the name of that gene with the use of the getAreaFromGeneNames function and org.Hs.eg.db. You are also able to specify strand information and infer genotypes from allele. Most notably of them all, the AllelicImbalance package allows you to also examine any potential mapping bias by analyzing the reference and alternate allele counts. Mapping bias is considered one of the major sources of error for ASE analysis packages and being able to determine whether mapping bias is present, and to what extent, is a major benefit for this package. All of these features give the AllelicImbalance package an advantage when compared to others. There are also some limitations to the AllelicImbalance package, and to other ASE analysis packages. For instance, all of the packages mentioned will return a chi square or P value to signify whether ASE is present, but none of the packages will show which allele is being over or under expressed. Furthermore, the statistical test used is highly dependent on the quality of the dataset as errors in counting the number of reads and mapping bias can cause a false positive or false negative result. Granted, this is the case with all statistical tests. There is also the issue of duplicate reads and how ASE analysis packages can handle them. It was found that removing duplicate reads can help lower the number of false positive ASE results while not altering the true ASE sites (Castel et al., 2015). However, the AllelicImbalance package does not remove any duplicate reads. With that being said, duplicate reads can often be filtered out both prior to and after mapping the chosen dataset. Finally, the AllelicImbalance package, and other ASE analysis packages within R, only function with heterozygous sites. With all these limitations in mind, there is a clear path forward for improvement within the AllelicImbalance package, and other ASE analysis tools within R.

Dataset:

For the AllelicImbalance package, you are required to use RNA sequence data which has been mapped to a reference genome and can be in either BAM, VCF or BCF format. For the purposes of this tutorial, the sample dataset provided with the AllelicImbalance package will be used (Gadin et al., 2015). This data is originally from a 60-individual sample of a paired end RNA sequence data CEU HapMap project, accession number: PRJEB2047, which has been used across multiple studies (Bai et al., 2014) (Kim & Pourmand, 2013) (Gabriele et al., 2017). The reasoning behind choosing this dataset for the tutorial is that everyone who installs the AllelicImbalance package in R will have access to it. Furthermore, this dataset contains information for every aspect of the package, so for this tutorial, it is the best choice as to explain every feature. Other datasets can still be used as long as they are in a BAM, VCF or BCF format, as mentioned previously.

Manual:

Note: This manual is an R script which can be downloaded from this link:

https://studentuml-my.sharepoint.com/:u:/g/personal/nicholas_hebert_student_uml_edu/EZGaHAKu6YBBget2E6rffBgBfshdF1oiPg1kx3iPlV1Ufw?e=HHVnlF

```
# First, we must Install and load the AllelicImbalance package:

# Installing:

BiocManager::install("AllelicImbalance")

# Loading:

library(AllelicImbalance)

#Creating base variables for the ASE object:

# Identifying the path of the dataset:

Path <- system.file("extdata/ERP000101_subset",

                package = "AllelicImbalance")

# Note: This path is for the dataset included within the AllelicImbalance package.

# For other datasets, you would simply write the path to the file within your computer.

# Example:

Path <- "~/Downloads/Project/BAM"

# Now we need to specify the region we are going to analyze:

Region <- GRanges(seqnames = c("17"), ranges = IRanges(79478301, 79478361))
```

```
# Once the region and path are stored, we can obtain the reads from the region
# specified within the file specified:
Reads <- impBamGAL(Path, Region, verbose = FALSE)
# Since AllelicImbalance only functions for heterozygous sites, we need
# to obtain all heterozygotes within the sample:
Hetero <- scanForHeterozygotes(Reads, verbose = FALSE)
# Once we have all the required reads, we can obtain the allele counts:
Count <- getAlleleCounts(Reads, Hetero, verbose = FALSE)
# Using these variables to create the ASE object:
Object <- ASEsetFromCountList(Hetero, Count)
# Technically, we can do a chi square statistical test on this variable,
# however it is still incomplete.
# We will be using a significance level of 0.05 throughout this tutorial,
# so any value below that cutoff is significant.
chisq.test(Object)
# For some datasets we can add in the strand information, which is the case for this one:
Positive <- getAlleleCounts(Reads, Hetero, strand = "+", verbose = FALSE)
Negative <- getAlleleCounts(Reads, Hetero, strand = "-", verbose = FALSE)
# Now we can recreate the ASESetList variable using stranded information:
ObjectWithStrands <- ASEsetFromCountList(Hetero, countListPlus = Positive
                                , countListMinus = Negative)
# We can then run chi square on the stranded variable:
chisq.test(ObjectWithStrands)
```

```
# We are also able to add phenotype data:

Phenotype <- 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=""))

# With this phenotype data, we can create a new ASESetList variable:

ObjectWithPhenotype <- ASEsetFromCountList(Hetero, Count, colData = Phenotype)

# Additionally, we can add in reference and alternate allele information:

# Adding in reference alleles:

ref(Object) <- c("G", "T", "C")

# Inferring alternate alleles:

Alternate <- inferAltAllele(Object)

alt(Object) <- Alternate

# You can also add reference alleles based on a reference genome:

data("ASEset.sim")

RefGen <- system.file('extdata/hg19.chr17.subset.fa', package='AllelicImbalance')

Reference <- refAllele(ASEset.sim,fasta=RefGen)

ref(ASEset.sim) <- Reference

# We now have our reference and alternate alleles noted.

# After noting the reference and alternate alleles, we are able to add or infer a genotype:

genotype(Object) <- inferGenotypes(Object)

# Test to make sure it worked:
```

```
genotype(Object) [,1:4]

# Next up we need to infer the phase information. This can most easily be done

# using a VCF file and the VariantAnnotation package.

# First, we must install the VariantAnnotation package:

BiocManager::install("VariantAnnotation")

# Now we have to load the package:

library(VariantAnnotation)

# Time to get started on creating a matrix to hold the phase information:

set.seed(1)

rows <- nrow(Object)

cols <- ncol(Object)

Phase1 <- matrix(sample(c(1,0), replace=TRUE, size=rows*cols), nrow=rows, ncol=cols)

Phase2 <- matrix(sample(c(1,0),replace=TRUE, size=rows*cols), nrow=rows, ncol=cols)

Phase.Materials <- matrix(paste(Phase1, sample(c("|","|","/"), size=rows*cols, replace=TRUE),

                        Phase2, sep=""), nrow=rows, ncol=cols)

phase(Object) <- Phase.Materials

# Now that the matrix is established, we can load in the VCF:

VCF <- system.file("extdata/ERP000101_subset/ERP000101.vcf", package =

"AllelicImbalance")

Phase <- readGT(VCF)

# We can now subset and order:

Object_Subbed <- Object[ ,colnames(Object) %in% colnames(Phase)]

Phase_Subbed <- Phase[ ,colnames(Phase) %in% colnames(Object_Subbed)]
```

```
Phase_Ordered <- Phase_Subbed[ ,match(colnames(Object_Subbed),
colnames(Phase_Subbed))]
# Now that we have the extra categories filled out, we can perform the tests again:
# For chi square:
chisq.test(ObjectWithStrands[,5:8], "-")
# This test is showing the values for rows 5-8 of the minus strand.
# In order to use a binomial test, we must set reference alleles for the stranded object as well:
ref(ObjectWithStrands) <- c("G", "T", "C")
binom.test(ObjectWithStrands, "-")
# If we want to use summary functions with this data, it would be:
Area <- granges(Object)
Area
# That is the end as far as analyzing ASE goes for the AllelicImbalance package.
# However, we are also able to analyze any potential mapping bias, which is one of the
# major sources of error for analyzing ASE.
# In order to do this, we must first create an example count list:
# NOTE: MUST FIRST HAVE REFERENCE AND ALTERNATE ALLELES
set.seed(1)
CountList <- list()
# Now that the list is created, we can start to create a sample of SNPs:
set.seed(1)
countListUnknown <- list()
samples <- paste(rep("sample",10),1:10,sep="")
```

```
snps <- 1000

for(snp in 1:snps){

  count<-matrix(0, nrow=length(samples), ncol=4,

                dimnames=list(samples, c('A','T','G','C')))

  alleles <- sample(1:4, 2)

  for(sample in 1:length(samples)){

    count[sample, alleles] <- as.integer(rnorm(2,mean=50,sd=10))

  }

  countListUnknown[[snp]] <- count

}

# Now we can create row ranges for the data:

Grange <- GRanges(seqnames = Rle(sample(paste("chr",1:22,sep=""),snps, replace=TRUE)),

                  ranges = IRanges(1:snps, width = 1,

                                   names= paste("snp",1:snps,sep="")),

                  strand="*")

# Now we can create a new ASESetList object:

Mapping_Bias <- ASEsetFromCountList(Grange, countListUnknown = countListUnknown)

# Adding a radnom reference allele:

ref(Mapping_Bias) <- randomRef(Mapping_Bias)

# Adding a genotype:

genotype(Mapping_Bias) <- inferGenotypes(Mapping_Bias)

# Now we have to obtain the reference fraction:

ReferenceFraction <- fraction(Mapping_Bias, top.fraction.criteria = "ref")
```

```
# Checking the mean to see if there is any mapping bias:

mean(ReferenceFraction)

# As we can see, the mean is 0.5003092, which means that there is little to no mapping bias.

# We know this because we expect that the alleles will be expressed equally, which means

# 0.5 for each allele.
```

Video link:

https://studentuml-my.sharepoint.com/:v:/g/personal/nicholas_hebert_student_uml_edu/ERqiKCe6PhRAn_-4obOhkEIBX29lC8CjGmBtrOKhaT2EaQ?e=3Zjiv2

Resources:

1) Jesper Robert Gådin, Ferdinand Matthijs van't Hooft, Per Eriksson, Lasse Folkersen (2015). "AllelicImbalance: an R/bioconductor package for detecting, managing, and visualizing allele expression imbalance data from RNA sequencing. BMC Bioinformatics." *BMC Bioinformatics*, **16**(194). doi: 10.1186/s12859-015-06202, http://dx.doi.org/10.1186/s12859-015-0620-2.
2) Morgan M, Pagès H, Obenchain V, Hayden N (2020). *Rsamtools: Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import*. R package version 2.6.0, https://bioconductor.org/packages/Rsamtools.
3) Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10(3):R25 (2009).
4) Curia, M. C., De Iure, S., De Lellis, L., Veschi, S., Mammarella, S., White, M. J., et al. (2012). Increased variance in germline allele-specific expression of APC associates with colorectal cancer. *Gastroenterology* 142, 71.e1–77.e1. doi: 10.1053/j.gastro.2011.09.048
5) Gaur, U., Li, K., Mei, S. *et al.* Research progress in allele-specific expression and its regulatory mechanisms. *J Appl Genetics* **54,** 271–283 (2013). https://doi.org/10.1007/s13353-013-0148-y
6) Valle, L., Serena-Acedo, T., Liyanarachchi, S., Hampel, H., Comeras, I., Li, Z., et al. (2008). Germline allele-specific expression of TGFBR1 confers an increased risk of colorectal cancer. *Science* 321, 1361–1365. doi: 10.1126/science.1159397
7) Castel, S.E., Levy-Moonshine, A., Mohammadi, P. *et al.* Tools and best practices for data processing in allelic expression analysis. *Genome Biol* **16,** 195 (2015). https://doi.org/10.1186/s13059-015-0762-6
8) Mayba O, Gilbert H (2020). *MBASED: Package containing functions for ASE analysis using Meta-analysis Based Allele-Specific Expression Detection*. R package version 1.24.0.
9) Jiaxin Fan, Jian Hu, Chenyi Xue, Hanrui Zhang, Muredach P. Reilly, Rui Xiao, Mingyao Li (2019). ASEP: gene-based detection of allele-specific expression in a population by RNA-seq. bioRxiv. https://doi.org/10.1101/798124
10) Bai, Y., Ni, M., Cooper, B., Wei, Y., & Fury, W. (2014). Inference of high resolution HLA types using genome-wide RNA or DNA sequencing reads. *BMC genomics*, *15*(1), 325. https://doi.org/10.1186/1471-2164-15-325
11) Kim, H. J., & Pourmand, N. (2013). HLA typing from RNA-seq data using hierarchical read weighting [corrected]. *PloS one*, *8*(6), e67885. https://doi.org/10.1371/journal.pone.0067885
12) Gabriele M, Vulto-van Silfhout AT, Germain PL, et al. YY1 Haploinsufficiency Causes an Intellectual Disability Syndrome Featuring Transcriptional and Chromatin Dysfunction. American Journal of Human Genetics. 2017 Jun;100(6):907-925. DOI: 10.1016/j.ajhg.2017.05.006.
13) Obenchain V, Lawrence M, Carey V, Gogarten S, Shannon P, Morgan M (2014). "VariantAnnotation: a Bioconductor package for exploration and annotation of genetic variants." *Bioinformatics*, **30**(14), 2076-2078. doi: 10.1093/bioinformatics/btu168.
14) Yonghong Li, Andrew Grupe, Charles Rowland, Petra Nowotny, John S.K. Kauwe, Scott Smemo, Anthony Hinrichs, Kristina Tacey, Timothy A. Toombs, Shirley Kwok, Joseph Catanese, Thomas J. White, Taylor J. Maxwell, Paul Hollingworth, Richard Abraham, David C. Rubinsztein, Carol Brayne, Fabienne Wavrant-De Vrièze, John Hardy, Michael