

asSeq: A set of tools for the study of allele-specific RNA-seq data

Wei Sun and Vasyl Zhabotynsky

February 22, 2013

1 Overview

```
> library(asSeq)
```

This vignette describes how to use R/asSeq to perform eQTL mapping using total expression and/or allele-specific expression, including a pipeline for input data preparation.

2 A brief Introduction

RNA-seq has become one of the major solutions for genome-wide inquiry of transcriptome variation. Allele-specific expression (ASE), which can be measured by RNA-seq but not by traditional microarray, provides a new perspective of transcriptome variation. Allelic imbalance of gene expression may be due to *cis*-acting genetic variant or parent-of-origin regulation. Currently this R package `asSeq` only provides support for assessing *cis*-acting eQTL. Later we will include another set of functions that can dissect (*cis*-acting) genetic effect and parent-of-origin effect. Here *cis*-acting regulation means a genetic variant on one allele (i.e., the maternal or paternal allele) only affects the expression on the same allele. A *cis*-eQTL often refers to a local-eQTL. In this document, we use a more precise definition of *cis*-eQTL such that they are the eQTL associated with ASE. Interested readers are referred to Sun (2102) [1] and Sun and Hu (2012) [2] for more details. Sun (2102) [1] provides the details of a statistical method for eQT mapping using both total expression and ASE. Sun and Hu (2012) [2] is a review paper that reviews topics related with eQTL mapping using ASE or isoform-specific expression. In addition to main functions for eQTL mapping using total expression or ASE, `asSeq` also provides utility functions for quality control or extracting allele-specific RNA-seq reads, which are the RNA-seq reads that overlap with heterozygous genetic markers. Figure 1 shows a complete pipeline of eQTL mapping using ASE. Some additional scripts are provided for certain steps of this pipeline.

3 DNA Data Processing

We assume DNA genotype data are available, either from SNP array or DNA sequencing. It is possible to call SNPs or indels in exonic regions from RNA-seq data, but we do not consider such approach here. First we need to impute haplotype data in a larger number of SNPs, for example, the SNPs/indels from the 1000 Genome Project. Large number of SNPs/indels provide higher chance that a RNA-seq read overlaps with a heterozygous genetic marker, hence is identified as allele-specific read. Haplotypes are necessary for eQTL mapping and estimation of gene level ASE, which is defined by counting the total number of allele-specific reads that may overlap with different SNPs/indels within a gene.

A set of R scripts are provided for haplotype phasing using MACH in folder `scripts/MACH` of this R package. In these R scripts, we start with the SNP genotype from Affymetrix 6.0 array (less than 800,000 autosome SNPs), and impute the haplotypes on more than 36.8 million autosome SNPs from the 1000 Genome Project. Software MACH and all the 1000 Genome Project data needed for phasing can be downloaded from <http://www.sph.umich.edu/csg/abecasis/MACH/download/>. Briefly, it takes four steps to obtain haplotype estimates.

1. `step1_prepare_for_MACH.R`: prepare genotype data in merlin format as input files of MACH.

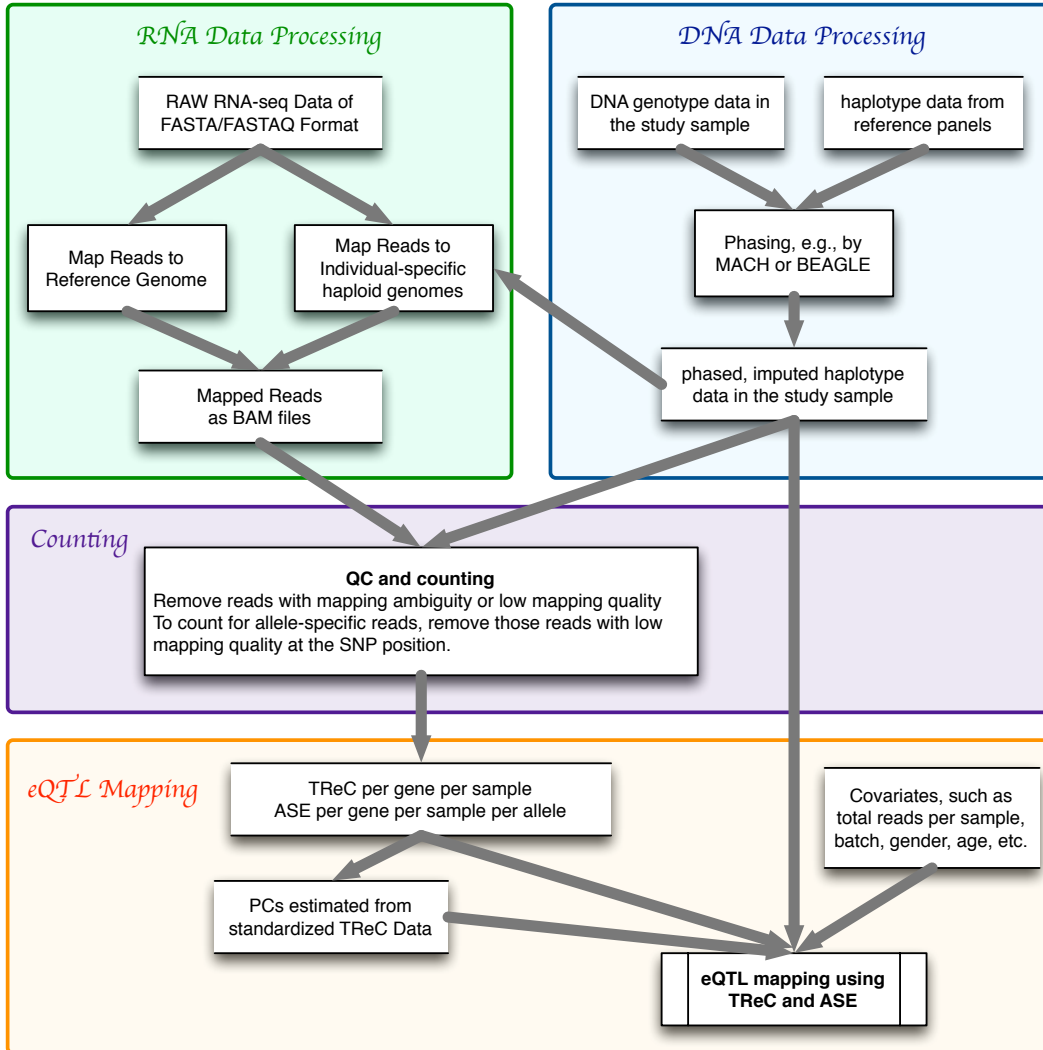


Figure 1: A pipeline for eQTL mapping using allele-specific RNA-seq data.

2. `step2_split_ref.R`: split reference haplotype files from 1000 Genome Project. We only use the data of European Ancestor individuals for this example.
3. `step3_run_MACH.R`: run MACH using parallel computation for each split reference file.
4. `step4_ligate.R`: ligate the MACH results

4 RNA Data Processing

In most of the studies, RNA data processing mainly refer to RNA-seq mapping. The commonly used strategy is to map the RNA-seq reads of all the individuals to the same reference genome. However, increasing evidence has shown that an alternative approach, to map the RNA-seq reads of each individual to its own haploid genomes, is more desirable in terms of identifying allele-specific reads. Certainly the two haploid genomes of a diploid individual are not known in most of the cases. However they can be estimated (sometimes referred to as pseudo-genome) by inserting phased indels and SNPs into the reference genome. These indels and SNPs have to be phased and it is often safe to assume the phasing is accurate within a gene. Possible switch errors of phasing in a longer range can be handled by specifically designed statistical methods in the association testing step. This is a method that we are currently working on.

5 QC and Counting

5.1 Prepare the BAM files

The following codes provide some simple steps for BAM file preparation. Samtools is required for steps 1, 2, and 4. In the 1st and 4th step, we count the number of reads in the original bam file and the processed bam file. In the 2nd step, we sort the bam file by read names. In the 3rd step, we use an R function `prepareBAM` of our R package `asSeq` to obtain uniquely mapped reads and apply a number of quality control criteria. The function `prepareBAM` not only excludes those reads that fail QC, but also fixes the annotation in the bam file, particularly the `FLAG` field of each record. For example, if one read is deleted, the `FLAG` field of its paired mate should be modified to reflect this change. `prepareBAM` can also do the sorting step, however we found samtools can often perform the sorting step faster with smaller amount of memory.

```
# -----  
# 1. counting  
# -----  
sami = "sample1"  
bami = "sample1_lane1_xyz.bam"  
ctF = sprintf("_count/count_%s.txt", sami)  
cmd1 = sprintf("samtools view %s | wc -l >> %s\n", bami, ctF)  
system(cmd1)  
  
# -----  
# 2. sorting  
# -----  
cmd2 = sprintf("samtools sort -n %s %s_sorted_by_name", bami, sami)  
system(cmd2)  
bamF = sprintf("%s_sorted_by_name.bam", sami)  
  
# -----  
# 3. getUnique and filtering  
# -----  
prepareBAM(bamF, sprintf("%s_sorted_by_name", sami), sortIt=FALSE)  
  
# -----  
# 4. counting again  
# -----  
cmd3 = sprintf("samtools view %s_sorted_by_name_uniq_filtered.bam | wc -l >> %s\n", sami, ctF)  
system(cmd3)
```

5.2 Obtain the list of heterozygous SNPs of each individual (the “snpList” file)

Given haplotype data of each individual generated in the previous step, we will generate a list of heterozygous SNPs to be used to identify allele-specific RNA-seq reads, which are the RNA-seq reads that overlap with at least one heterozygous SNP. We refer to this file as a “snpList” file. The first a few rows of a “snpList” file look like the following:

```
chr1 1019668 G A  
chr1 1020428 T C  
chr1 1020496 A G  
chr1 1029889 C T
```

The four columns are chromosome, position, allele 1 (the allele on haplotype 1) and allele 2 (the allele on haplotype 2), respectively. The 3rd and 4th columns are essentially the two haplotypes of this individual. Which haplotype is the 1st/2nd haplotype of an individual can be arbitrarily defined. However, such definition should remain the same throughout the analysis. This snpList file will be used by R function

`extractAsReads` to extract allele specific reads. Only the SNPs of heterozygous genotypes are needed in this file, although the function `extractAsReads` can automatically filter out those SNPs with homozygous genotypes, if there is any. An R code (`get_snpList.R`) that can generate such a `snpList` file can be in folder `scripts/snpList` of this R package.

5.3 Extract the allele-specific reads

R function `extractAsReads` from our R package `asSeq` can be used to extract allele-specific sequence reads. For example, the command can be

```
extractAsReads(input, snpList, outTag="myOutput", prop.cut=.5, min.snpQ=10, phred=33)
```

Three main parameters are:

- `input`: input file name, this should be file of BAM format
- `snpList`: name of the file including a SNP list, must be tab-delimited file with four columns, chromosome, position, allele 1 and allele 2, without header
- `outputTag`: output files will be named as `outputTag_hap1.bam`, `outputTag_hap2.bam`, and `outputTag_hapN.bam`.

There are several additional parameters, and three relatively more important ones are:

- `prop.cut`: one RNA-seq read may overlap with multiple heterozygous SNPs. We assign a read to one of the two haplotypes if the proportion of those heterozygous SNPs suggesting the read is from that haplotype is larger than `prop.cut`.
- `min.snpQ`: If the sequencing quality of the base-pair at a SNP is smaller than `min.snpQ`, we will not use this SNP to call allele-specific read. The default value is 10, which means 10 above the baseline phred score specified by parameter `phred`.
- `phred`: baseline phred score. Use 33 for Illumina 1.8+, and use 64 for sequences from previous Illumina pipeline. see http://en.wikipedia.org/wiki/FASTQ_format for more details.

5.4 Counting the total number of reads and the number of allele-specific reads

We count the number of reads per exon set, which includes one or more exons. A single or paired-end read overlaps with an exon set if it overlaps with all the exons in this exon set and it does not overlap with any other exon. In the existing annotations, the exons are often overlapping. We produce the annotation of non-overlapping exons for mouse (`Mus_musculus.NCBIM37.67_data.zip`) and human (`Homo_sapiens.GRCh37.66_data.zip`) in bed format, which can be downloaded from http://www.bios.unc.edu/~weisun/software/isoform_files/. The scripts used to generate (and check) these non-overlapping exons are included in this R package at folder `scripts/getUniqExon`.

The actual counting step can be done in different ways. We modify the `intersectBed` function of `bedtools` to appropriately handle paired-end RNA-seq reads and implement it as an R function `countReads` of another R package named `isoform`.

```
library(isoform)
countReads(bamFile, bedFile, outFile)
```

where `bamFile` is the name of the bam file after sorting by read name and after the QC steps, `bedFile` is the name of the annotation file of non-overlapping exons, and `outFile` is the name of output file, which is a text file that looks like the following:

```
17 chr10_100|ENSG00000107651|10;
 3 chr10_100|ENSG00000107651|8;chr10_100|ENSG00000107651|10;
 2 chr10_100|ENSG00000107651|8;chr10_100|ENSG00000107651|9;chr10_100|ENSG00000107651|10;
11 chr10_100|ENSG00000107651|9;chr10_100|ENSG00000107651|10;
```

Each line of this file is the count of the number of RNA-seq reads overlapping a particular exon set. For example, the 2nd line of the above example says that there are 3 RNA-seq reads that overlap with the exon 8 and 10 of gene cluster chr10_100, and these two exons are part of gene ENSG00000107651. A gene cluster includes one or more genes that share at least one exon. This output file provides enough information for the study of gene expression at isoform level. However, we will not study RNA isoform expression here. User can easily collapse these counts at exon set level into gene level. If some exons are shared by two genes, we will treat those exons shared by two genes as a gene.

Allele-specific expression has been extracted into two BAM files in the previous step and they can be counted the same way.

6 eQTL mapping

6.1 Obtain genotype data for eQTL mapping

Denote the 1st and 2nd haplotype of an individual by \mathbf{h}_1 and \mathbf{h}_2 , respectively. As mentioned before, which haplotype is the 1st/2nd haplotype of an individual can be arbitrarily defined. However, such definition should be consistent with those used in the “snpList” file. For the k -th SNP, denote the alleles on haplotypes \mathbf{h}_1 and \mathbf{h}_2 as h_{1k} and h_{2k} , respectively. We define a “phased” genotype for the k -th SNP as $h_{1k}h_{2k}$. Then if a SNP has two alleles, A and B, the phased genotype can be AA, AB, BA or BB. In contrast to common definition of genotype, AB and BA are distinct phased genotypes. The genotype data needed for eQTL mapping should be a data matrix with each row for an individual and each column for a SNP. Each entry in the matrix is 0, 1, 3, or 4, corresponding to genotype AA, AB, BA, or BB. Which allele of a SNP is A or B can be defined arbitrarily. For example, we can consider the B allele as the minor allele.

6.2 eQTL mapping

Finally (yes, finally...), we obtain all the data needed for eQTL mapping using both total expression and allele-specific expression by the R function `treCase`:

```
treCase(Y, Y1, Y2, X, Z, output.tag, p.cut)
```

These parameters/input data are explained in the following. We also explain which of the previous steps generate such input data.

- **Y**: matrix of gene expression in terms of total read count (TReC). Each row corresponds to one sample and each column corresponds to one gene. This data is generated at step 5.4.
- **Y1/Y2**: matrices of allele-specific gene expression. Each row corresponds to one sample and each column corresponds to one gene. Specifically, Y1 and Y2 are the expression from first and second haploid genome, respectively. The first and second haploid genome is defined as the column order in snpList file, which is generated at step 5.2. Y1 and Y2 are generated at Step 5.4.
- **X**: matrix of confounding covariates. Each row corresponds to one sample and each column corresponds to one variable. Usually these covariates include $\log(\text{total number of reads per sample})$ and PCs obtained from \log transformed and total-number-of-reads-per-sample corrected expression matrix.
- **Z**: matrix of phased genotype data. Each row corresponds to one sample/individual and each column corresponds to one SNP. Z must takes value of 0 (AA), 1 (AB), 3 (BA) or 4 (BB), where A and B are two alleles of a SNP. AB and BA are different since they represent phased genotypes. Specifically, AB means the 1st/2nd haplotype harbors the A/B allele, respectively, while BA means the 1st/2nd haplotype harbors the B/A allele, respectively. This file is generated at step 6.1.
- **output.tag** The results of eQTL computation will be output into two files: `output.tag_eqtl.txt` and `output.tag_freq.txt`. The former file includes (gene, SNP) pairs and their corresponding p-values, and the latter file includes the distribution of all eQTL p-values, which may be useful to calculate FDR.

- `p.cut` p-value cut-off, only the eQTL associations with p-value smaller than `p.cut` are saved.

Joint model of both total expression and allele-specific expression will only be attempted if there are enough allele-specific reads in enough samples, which are specified by parameters `min.AS.reads` and `min.AS.sample`. We have implemented a likelihood ratio test (with p-value cutoff specified by `transTestP`) to assess whether total read count and allele-specific read count give consistent association strength. If the answer is affirmative, joint model p-value will be reported as final p-value, otherwise the p-value from total read count will be reported as the final p-value.

The columns of the output files include

- `GeneRowID`, `MarkerRowID` : row ID for gene and DNA marker, respectively.
- `TReC_b`, `TReC_Chisq`, `TReC_df`, `TReC_Pvalue`: regression coefficient, Chi-square test statistic, degree of freedom, and p-value of TReC test. Similarly we have these columns for the results of ASE model and Joint model.
- `n_TReC`, `n_ASE`: sample size for TReC and ASE model.
- `n_ASE_Het`: among those samples with ASE data, how many have heterozygous genotype on the DNA marker.
- `trans_Chisq`, `trans_Pvalue`: test statistic and p-value to assess whether the eQTL is a trans-eQTL. Small p-value implies trans-eQTL.
- `final_Pvalue`: `final_Pvalue` equals to `TReC_Pvalue` if it is a trans-eQTL, and it equals to `Joint_Pvalue` otherwise.

There are two other functions for eQTL mapping using Total Read Count (TReC) only (function `trec`)

```
trec(Y, X, Z, output.tag, p.cut)
```

or allele-specific expression only (function `ase`).

```
ase(Y1, Y2, Z, output.tag, p.cut)
```

In most situations, these two functions are not necessary because function `trecase` already output the test p-values using total read count only or ASE only. Function `trec` only uses total read count `Y`, but does not need allele-specific read count `Y1` and `Y2`. The genotype data for function `trec` (parameter `Z`) should be numerical coding of genotype (e.g., additive coding with 0, 1, and 2 for AA, AB, and BB, respectively), rather than phased genotype. Function `ase` only uses allele-specific reads `Y1` and `Y2` and the genotype data `Z` should be the phased genotype, the same as the `Z` used by function `trecase`.

If one only keep the most significant eQTL for each gene (e.g., this is reasonable for a local eQTL scan), an ideal solution of multiple testing correction is to evaluate the permutation p-value of the most significant eQTL for each gene, and then choose a permutation p-value cutoff to control FDR across all the genes. Permutation test is computationally intensive. Larger number of permutations is only needed for significant associations. We implement this strategy in function `trecaseP`. One example of using this R function is listed as below.

```
trecaseP(Y, Y1, Y2, X, Z, np.max=5000, np=c(20, 100, 500, 1000, 2500),
        aim.p=c(0.5, 0.2, 0.1, 0.05, 0.02), confidence.p=0.01)
```

The parameters `Y`, `Y1`, `Y2`, `X`, `Z` are the same as those in function `trecase`. The extra parameters specify how to do permutations. In this example, we will do at most 5000 permutations (`np.max=5000`). However, we will terminate the permutation if at the number of permutations 20, 100, 500, 1000, and 2500 (`np=c(20, 100, 500, 1000, 2500)`), we already have >99% of confidence (`confidence.p=0.01`) that the permutation p-value is larger than 0.5, 0.2, 0.1, 0.05, and 0.02 (`aim.p=c(0.5, 0.2, 0.1, 0.05, 0.02)`), respectively.

References

- [1] Wei Sun (2012) A statistical framework for eQTL mapping using RNA-seq data. *Biometrics*. **68**(1):1-11.
- [2] Wei Sun and Yijuan Hu (2012), eQTL mapping using RNA-seq data, *Statistics in Biosciences*. in press