Detection of Differentially Allelic Expressed Regions (DAERs) by hmmASE

Jing Xie

Introduction

The package *hmmASE* performs detection of differentially allelic expressed regions (DAERs) by hmmASE method. Depending on the if there exhibits allelic specific expression in the normal group, and whether if the allelic preference for two groups are different, there would be 9 possible states for each SNP, i.e., B-Neutral, B-Hyper, B-hypo, M-Neutral, M-Hyper, M-hypo, P-Neutral, P-Hyper, P-hypo. The letter "B", "M" and "P" represents "Balanced", "Maternal preference" and "Paternal preference" for the normal group respectively, wheras the term "Hyper","Hypo" and "Neutral" refers to the difference between the ASE ratios between abnorm and normal group. For example, a region or a single SNP cite which has state "M_hyper" means the proportion of gene expression comes from maternal allele is significantly greater than that of paternal allele in the normal group, and this proportion in the abnormal group is even greater than the normal group which means the abnormal group exhibits a stronger maternal preference. The common assupmption of the indepedence between SNPs is not realistic and is too simplistic to understand the complexity of the data. The hmmASE method is based on a bayesian hidden markov model thus can take into account the correlation between adjecent SNPs and gives a more accurate inference about the hidden states of each SNP. In this manual, we first illustarte the usage of this package by providing a quick start guide. Then a step-by-step example will show addtional functionlity and features, and provides more insights about this package.

Quick start guide

Given the read counts data from both alleles for the normal and abnormal groups, the detection of differentially allelic expressed regions (or SNP) can be performed by calling the function $ASE.HMM()$. Except for the specification of the raw data for each groups and each allele, users need to specify: (1) the number of biological replicates in their data by $rep = c(r_1, r_2)$; (2) whether or not to remove the outliers in the original data and also the cutoff to define outliers if $ex.rm = TRUE$; (3) the decoding approach for Hidden Markov model, global decoding or local decoding method; (4) the minimum length of the desired regions and minimum number of SNPs and maximum distance between two adjecent SNPs inside a region. Although we provide the option to remove the extreme data from the HMM model, we recommend careful attention to this option as well as the choice of the cutoff value, since the removal of a large part of data from the HMM sequence would affect the accuracy of states inferences.

```
library("hmmASE")
toy.norm.M<- read.csv(file="Yourdata.norm.M.csv")
toy.abnorm.M<- read.csv(file="Yourdata.abnorm.M.csv")
toy.norm.P<- read.csv(file="Yourdata.norm.P.csv")
toy.abnorm.P<- read.csv(file="Yourdata.abnorm.P.csv")

region.res<- ASE.HMM(norm.M = toy.norm.M,norm.P = toy.norm.P,abnorm.M =   toy.abnorm.M,abnorm.P = toy.al
                     min.length=0,min.SNP=1,max.dist=1e20,cutoff=6.9)
```

Example

In this section, we will show how to apply *hmmASE* package to perform DAERs detection step by step. The starting point is to read in the four raw data sets which contain the read counts from maternal and paternal allele for two groups. Note that the dataset need to be in the same structure as the toy datasets in this

package. Specifically, the first 4 columns should be Chromosome, GeneID, Gene_name and Position, and the last last a few columns would be the read counts data. If the information about chromosome, GeneID or Gene_name is not available, one may still need to include the empty column. The number of biological replicates can differ for two groups.

```
library("hmmASE")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data("toy.norm.M")
data("toy.abnorm.M")
data("toy.norm.P")
data("toy.abnorm.P")
head(toy.abnorm.M,n=10)
```

```
##    Chromosome      GeneID Gene_name Position M1 M2 M3 M4
## 1           1 XLOC_000955     CLIC6   123725  1  2  1 23
## 2           1 XLOC_000320     RCAN1   362318  0 NA 43 NA
## 3           1 XLOC_000320     RCAN1   362525 22 24  0 33
## 4           1 XLOC_000960   FAM165B   463616 11 24 10 22
## 5           1 XLOC_000960   FAM165B   466250  0 NA 64 NA
## 6           1 XLOC_000960   FAM165B   466306  0 NA 70 NA
## 7           1 XLOC_000960   FAM165B   472805 12 12 44 41
## 8           1 XLOC_000960   FAM165B   473958  1 NA NA  3
## 9           1 XLOC_000960   FAM165B   473996  1 NA 15  2
## 10          1 XLOC_000963     MRPS6   686950  1 NA  8  8
```

```
head(toy.abnorm.P,n=10)
```

```
##    Chromosome      GeneID Gene_name Position P1 P2 P3 P4
## 1           1 XLOC_000955     CLIC6   123725  0  0  3 24
## 2           1 XLOC_000320     RCAN1   362318 12 NA 36 NA
## 3           1 XLOC_000320     RCAN1   362525 18 16 61 25
## 4           1 XLOC_000960   FAM165B   463616  6 19 15 16
## 5           1 XLOC_000960   FAM165B   466250 98 NA 67 NA
## 6           1 XLOC_000960   FAM165B   466306 73 NA 97 NA
## 7           1 XLOC_000960   FAM165B   472805 12 16 40 55
## 8           1 XLOC_000960   FAM165B   473958  1 NA NA  0
## 9           1 XLOC_000960   FAM165B   473996  1 NA  3  1
## 10          1 XLOC_000963     MRPS6   686950  1 NA  4  5
```

```
head(toy.norm.M,n=10)
```

```
##    Chromosome      GeneID Gene_name Position M1 M2 M3 M4
## 1           1 XLOC_000955     CLIC6   123620  1  1  2  4
## 2           1 XLOC_000955     CLIC6   123906  1 NA NA  1
## 3           1 XLOC_000955     CLIC6   124021  0 NA NA NA
## 4           1 XLOC_000955     CLIC6   124223  1 NA  0  2
```

```
## 5           1 XLOC_000955     CLIC6   135862  1 NA  2  4
## 6           1 XLOC_000955     CLIC6   135961  5  5  1  2
## 7           1 XLOC_000955     CLIC6   137920  2 NA  0  0
## 8           1 XLOC_000955     CLIC6   178734  1 NA NA  4
## 9           1 XLOC_000955     CLIC6   178833  1 NA NA NA
## 10          1 XLOC_000955     CLIC6   178921  0  0 NA NA
```

```
head(toy.norm.P,n=10)
```

```
##    Chromosome      GeneID Gene_name Position P1 P2 P3 P4
## 1           1 XLOC_000955     CLIC6   123620  0  1  0  0
## 2           1 XLOC_000955     CLIC6   123906  0 NA NA  0
## 3           1 XLOC_000955     CLIC6   124021  1 NA NA NA
## 4           1 XLOC_000955     CLIC6   124223  0 NA  1  0
## 5           1 XLOC_000955     CLIC6   135862  2 NA  0  0
## 6           1 XLOC_000955     CLIC6   135961  2  0  0  0
## 7           1 XLOC_000955     CLIC6   137920  1 NA  1  1
## 8           1 XLOC_000955     CLIC6   178734  0 NA NA  0
## 9           1 XLOC_000955     CLIC6   178833  1 NA NA NA
## 10          1 XLOC_000955     CLIC6   178921  1  1 NA NA
```

The first step of *hmmASE* workflow is to apply the Haldane-Anscombe correction and then transform the raw read counts data into the format that the Hidden Markov model base on, i.e., the logit transformation of the ASE ratio in normal group O_1 and the difference of the ASE ratios for two groups in logit form O_2. The *data.prep* () function can do this job and we need to specify the number of replicates for each group by $rep = c(r_1, r_2)$ where r_1 and r_2 are the number of biological replicates for normal and abnormal group respectively.

```
rawinput<- data.prep(norm.M=toy.norm.M,norm.P=toy.norm.P,
                     abnorm.M=toy.abnorm.M,abnorm.P=toy.abnorm.P,rep=c(4,4))
head(rawinput,5)
```

```
##    Chromosome      GeneID Gene_name S Position           O1          O2
## 1           1 XLOC_000320     RCAN1 1   362525  0.613594549 -1.02316644
## 57          1 XLOC_000960   FAM165B 2   463616  0.009852296  0.16381120
## 58          1 XLOC_000960   FAM165B 3   466250 -0.952900477  0.01529996
## 59          1 XLOC_000960   FAM165B 4   466306 -0.050483905 -0.82849977
## 60          1 XLOC_000960   FAM165B 5   472805 -0.365934269  0.24715073
```

The second step is to estimate the initial value of the parameters by calling the *data.trans* () function.

```
rawinput_r<- data.trans(rawinput)
names(rawinput_r)
```

```
## [1] "inputHMM" "initial"
```

```
head(rawinput_r[[1]],5)
```

```
##    S Position           O1          O2   dist
## 1  1   362525  0.613594549 -1.02316644      0
## 57 2   463616  0.009852296  0.16381120 101091
## 58 3   466250 -0.952900477  0.01529996   2634
## 59 4   466306 -0.050483905 -0.82849977     56
## 60 5   472805 -0.365934269  0.24715073   6499
```

```
names(rawinput_r[[2]])
```

```
##  [1] "pi"       "c"        "m.plus"   "m.minus"  "mu.plus"  "mu.minus"
##  [7] "sigma"    "sigma1"   "sigma2"   "delta"    "delta1"   "delta2"
```

```
## [13] "rho"      "tran.p"
```

The third step is to update the parameters by the EM algorithm and find the best sequence of predicted hidden states by decoding the Hidden Markov model. There are two options of decoding method in this package, local decoding and global decoding (Viterbi algorithm).

```
res.rawinput<- EM(initial.value=rawinput_r[[2]],
                              obs=rawinput_r[[1]],decoding="Global")
```

```
## EM algorithm starts:
## Iteration  1
## Iteration  2
## Iteration  3
## Iteration  4
## Iteration  5
## Iteration  6
## Iteration  7
## Iteration  8
## Iteration  9
## Iteration  10
## Iteration  11
## Iteration  12
## Iteration  13
## Iteration  14
## Iteration  15
## Iteration  16
## Iteration  17
## Iteration  18
## Iteration  19
## Iteration  20
## Iteration  21
## Iteration  22
## Iteration  23
## Iteration  24
## Iteration  25
## Iteration  26
## Iteration  27
## Iteration  28
## Iteration  29
## Iteration  30
## Iteration  31
## Iteration  32
## Iteration  33
## Iteration  34
## Iteration  35
## Iteration  36
## Iteration  37
## Iteration  38
## Iteration  39
## Iteration  40
## Iteration  41
## Iteration  42
## Iteration  43
## Iteration  44
## Iteration  45
```

```
## Iteration  46
## Iteration  47
## Iteration  48
## Iteration  49
## Iteration  50
## Iteration  51
## Iteration  52
## Iteration  53
## Iteration  54
## Iteration  55
## Iteration  56
## Iteration  57
## Iteration  58
## Iteration  59
## Iteration  60
## Iteration  61
## Iteration  62
## Iteration  63
## Iteration  64
## Iteration  65
## Iteration  66
## Iteration  67
## Iteration  68
## Iteration  69
## Iteration  70
## Iteration  71
## EM algorithm converges.
```

```
names(res.rawinput)
```

```
## [1] "res" "par"
```

In the final step, we put together the information in the raw input data and the predicted hidden states, and then find the regions where there are one or more consecutive SNPs. By setting $min.length = 0$, $min.SNP = 1$, $max.dist = 10^{20}$, we are loosing the contraints of the desired regions. Therefore, both the regions contains multiple SNPs and the single SNP cite which has different ASE status as its neighbor SNPs would be reported. Users may adjust the constraints accordingly.

```
RES<- res.rawinput$res
RES<- RES[order(RES$Position),]
region.in<- RES[,c(2,3,4,6)]
region.in<- merge(rawinput,region.in,by=c("Position","O1","O2"))
region.in<- region.in[order(region.in$Position),c(5:7,1:3,8)]
region.in$dist<- c(0,diff(region.in$Position))
head(region.in,5)
```

```
##         GeneID Gene_name S Position           O1          O2 state   dist
## 88 XLOC_000320     RCAN1 1   362525  0.613594549 -1.02316644     7      0
## 89 XLOC_000960   FAM165B 2   463616  0.009852296  0.16381120     7 101091
## 90 XLOC_000960   FAM165B 3   466250 -0.952900477  0.01529996     7   2634
## 91 XLOC_000960   FAM165B 4   466306 -0.050483905 -0.82849977     7     56
## 92 XLOC_000960   FAM165B 5   472805 -0.365934269  0.24715073     7   6499
```

```
Joint.region<- Region.Infer(EM.out=region.in,
                            min.length=0,
                            min.SNP=1,
```

```
                                max.dist=1e20
                                )$region
head(Joint.region,5)
```

```
##   region.cnt region.start region.end region.state num.SNP length
## 1          1       362525     687602    B-Neutral       9 325077
## 2          2       689562     694174      P-Hyper       2   4612
## 3          3       700848    1089173    B-Neutral      32 388325
## 4          4      1214806    1214806      P-Hyper       1      0
## 5          5      1215115    1254533    B-Neutral       6  39418
```