ExomeDepth

Vincent Plagnol

August 23, 2012

Contents

1 What ExomeDepth does and tips for QC

1.1 What ExomeDepth does and does not do

ExomeDepth uses read depth data to call CNVs from exome sequencing experiments. A key idea is that the test exome should be compared to a matched aggregate reference set. This aggregate reference set should combine exomes from the same batch and it should also be optimized for each exome. It will certainly differ from one exome to the next.

Importantly, ExomeDepth assumes that the CNV of interest is absent from the aggregate reference set. Hence related individuals should be excluded from the aggregate reference. It also means that ExomeDepth can miss common CNVs, if the call is also present in the aggregate reference. ExomeDepth is really suited to detect rare CNV calls (typically for rare Mendelian disorder analysis).

The ideas used in this package are of course not specific to exome sequencing and could be applied to other targeted sequencing datasets, as long as they contain a sufficiently large number of exons to estimate the parameters (at least 20 genes, say, but probably more would be useful). Also note that PCR based enrichment studies are often not well suited for this type of read depth analysis. The reason is that as the number of cycles is often set to a high number in order to equalize the representation of each amplicon, which can discard the CNV information.

1.2 Useful quality checks

Just to give some general expectations I usually obtain 150-280 CNV calls per exome sample (two third of them deletions). Any number clearly outside of this range is suspicious and suggests that either the model was inappropriate or that something went wrong while running the code. Less important and less precise, I also expect the aggregate reference to contain 5-10 exome samples. While there is no set rule for this number, and the code may work very well with fewer exomes in the aggregate reference set, numbers outside of this range suggest potential technical artifacts.

2 Create count data from BAM files

Firstly, to facilitate the generation of read count data, exon positions for the hg19 build of the human genome are available within `ExomeDepth`. This `exons.hg19` data frame can be directly passed as an argument of `getBAMCounts` (see below).

```
> library(ExomeDepth)
> data(exons.hg19)
> print(head(exons.hg19))

  chromosome start   end           name
1          1 12011 12058 DDX11L10-201_1
2          1 12180 12228 DDX11L10-201_2
3          1 12614 12698 DDX11L10-201_3
4          1 12976 13053 DDX11L10-201_4
5          1 13222 13375 DDX11L10-201_5
6          1 13454 13671 DDX11L10-201_6
```

To generate read count data, the function `getBamCounts` in `ExomeDepth` is set up to parse the BAM files. It generates an array of read count, stored in a GenomicRanges object. It is a wrapper around the function `countBamInGRanges.exomeDepth` which is derived from an equivalent function in the `exomeCopy` package. You can refer to the help page of `getBAMCounts` to obtain the full list of options. An example line of code (not evaluated here) would look like this:

```
> data(exons.hg19)
> my.counts <- getBamCounts(bed.frame = exons.hg19,
+                           bam.files = my.bam,
+                           referenceFasta = fasta)
```

`my.bam` is a set character vector of indexed BAM files. `fasta` is the reference genome in fasta format (only useful if one wants to obtain the GC content). `exons.hg19` are the positions and names of the exons on the hg19 reference genome (as shown above).

`getBAMCounts` creates an object of the GRanges class which can easily be converted into a matrix or a data frame (which is the input format for `ExomeDepth`). An example of GenomicRanges output generated by `getBAMCounts` is provided in this package (chromosome 1 only to keep the size manageable). Here is how this object could for example be used to obtain a more generic data frame:

```
> library(ExomeDepth)
> data(ExomeCount)
> ExomeCount.dafr <- as(ExomeCount[,  colnames(ExomeCount)], 'data.frame')
> ExomeCount.dafr$chromosome <- gsub(as.character(ExomeCount.dafr$space),
+                                    pattern = 'chr',
+                                    replacement = '')  ##remove the annoying chr letters
> print(head(ExomeCount.dafr))

  space start   end width           names        GC Exome1 Exome2 Exome3 Exome4
1     1 12012 12058    47 DDX11L10-201_1 0.6170213      0      0      0      0
2     1 12181 12228    48 DDX11L10-201_2 0.5000000      0      0      0      0
3     1 12615 12698    84 DDX11L10-201_3 0.5952381    118    242    116    170
4     1 12977 13053    77 DDX11L10-201_4 0.6103896    198     48    104    118
5     1 13223 13375   153 DDX11L10-201_5 0.5882353    516   1112    530    682
6     1 13455 13671   217 DDX11L10-201_6 0.5898618    272    762    336    372
  chromosome
1          1
2          1
3          1
4          1
5          1
6          1
```

3 Load an example dataset

We have already loaded a dataset of chromosome 1 data for four exome samples. We run a first test to make sure that the model can be fitted properly. Note the use of the subset.for.speed option that subsets some rows purely to speed up this computation.

```
> test <- new('ExomeDepth',
+              test = ExomeCount.dafr$Exome2,
+              reference = ExomeCount.dafr$Exome3,
+              formula = 'cbind(test, reference) ~ 1',
+              subset.for.speed = seq(1, nrow(ExomeCount.dafr), 100))
```

```
[1] 0.0229405
```

```
> show(test)
```

```
Number of data points:  266
Formula:  cbind(test, reference) ~ 1
Phi parameter (range if multiple values have been set):  0.0229405 0.0229405
Likelihood computed
```

4 Build the most appropriate reference set

Moving on toward a more useful computation, the first step is to select the most appropriate reference sample. This step is demonstrated below.

```
> my.test <- ExomeCount$Exome4
> my.ref.samples <- c('Exome1', 'Exome2', 'Exome3')
> my.reference.set <- as.matrix(ExomeCount.dafr[, my.ref.samples])
> my.choice <- select.reference.set (test.counts = my.test,
+                                    reference.counts = my.reference.set,
+                                    bin.length = (ExomeCount.dafr$end - ExomeCount.dafr$start)/1000,
+                                    n.bins.reduced = 10000)
```

```
[1] 0.006463586
[1] 0.004847499
[1] 0.004314528
```

```
> print(my.choice[[1]])
```

```
[1] "Exome2" "Exome1" "Exome3"
```

Using the output of this procedure we can construct the reference set.

```
> my.reference.selected <- apply(X = as.matrix( ExomeCount.dafr[, my.choice$reference.choice] ),
+                                MAR = 1,
+                                FUN = sum)
```

5 CNV calling

Now the following step is the longest one as the beta-binomial model is applied to the full set of exons:

```
> all.exons <- new('ExomeDepth',
+                  test = my.test,
+                  reference = my.reference.selected,
+                  formula = 'cbind(test, reference) ~ 1')
```

```
[1] 0.004514482
```

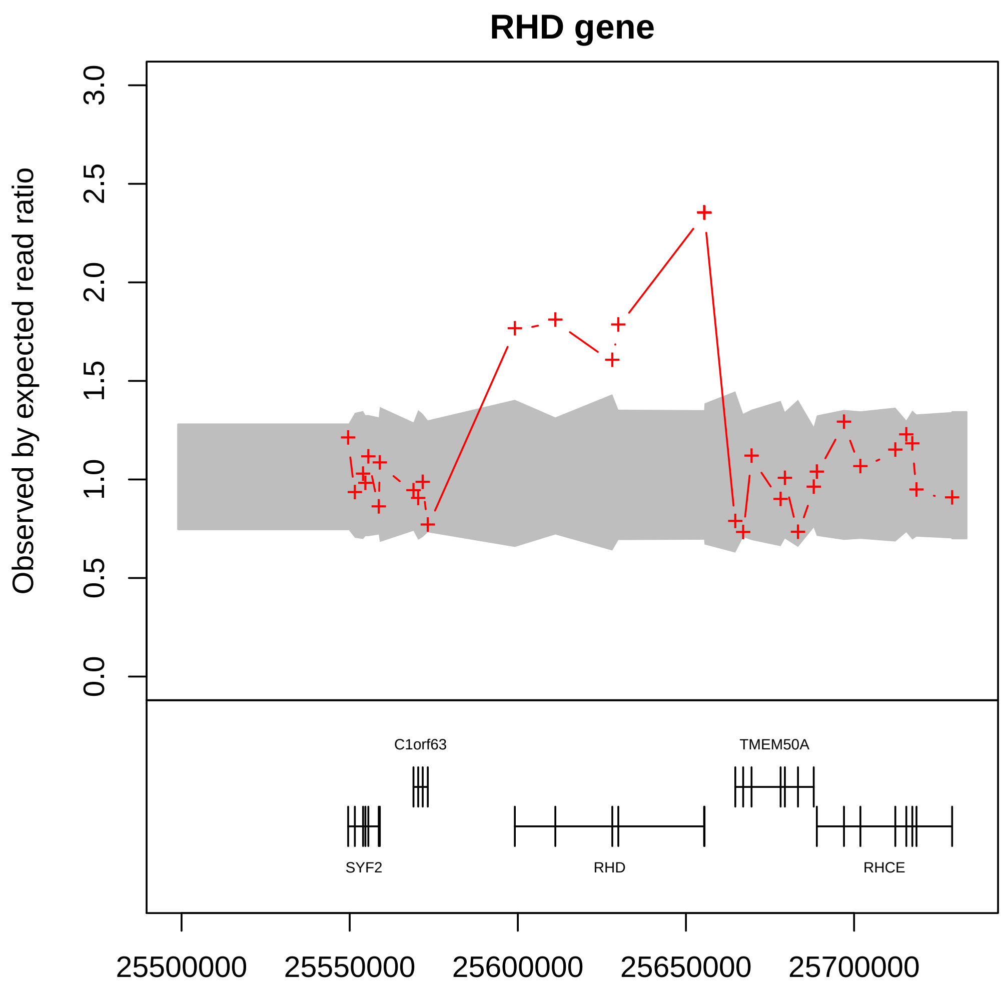
We can now call the CNV by running the underlying hidden Markov model:

```
> all.exons <- CallCNVs(x = all.exons,
+                       transition.probability = 10^-4,
+                       chromosome = ExomeCount.dafr$space,
+                       start = ExomeCount.dafr$start,
+                       end = ExomeCount.dafr$end,
+                       name = ExomeCount.dafr$names)

Number of hidden states: 3
Number of data points: 26547
Initializing the HMM
Done with the first step of the HMM, now running the trace back
Total number of calls: 23

> print(head(all.exons@CNV.calls))

  start.p end.p        type nexons    start      end chromosome
1      25    27    deletion      3    89553    91106          1
2      52    66    deletion     15   324290   523834          1
3     100   103 duplication      4   743956   745551          1
4     575   576    deletion      2  1569583  1570002          1
5     587   591    deletion      5  1592941  1603069          1
6    2324  2327    deletion      4 12976452 12980570          1
                     id    BF reads.expected reads.observed reads.ratio
1       chr1:89553-91106 12.40            224             68       0.304
2     chr1:324290-523834 13.40            380            190       0.500
3     chr1:743956-745551  7.67            201            336       1.670
4   chr1:1569583-1570002  5.53             68             24       0.353
5   chr1:1592941-1603069 13.90           1136            434       0.382
6 chr1:12976452-12980570 12.10            780            342       0.438
```

6 A visual example

The ExomeDepth object includes a plot function. This function shows the ratio between observed and expected read depth. The 95% confidence interval is marked by a grey shaded area. Here we use a common CNV located in the RHD gene as an example. We can see that the individual in question has more copies than the average (in fact two functional copies of RHD, which corresponds to rhesus positive).

```
> plot (all.exons,
+       sequence = '1',
+       xlim = c(25598981 - 100000, 25633433 + 100000),
+       count.threshold = 20,
+       main = 'RHD gene',
+       with.gene = TRUE)
```



7 Technical information about R session

```
> sessionInfo()

R version 2.15.1 (2012-06-22)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.iso885915       LC_NUMERIC=C
 [3] LC_TIME=en_US.iso885915        LC_COLLATE=C
 [5] LC_MONETARY=en_US.iso885915    LC_MESSAGES=en_US.iso885915
 [7] LC_PAPER=C                     LC_NAME=C
 [9] LC_ADDRESS=C                   LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.iso885915 LC_IDENTIFICATION=C

attached base packages:
[1] stats4    splines   stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
[1] ExomeDepth_0.8.4     Rsamtools_1.8.5      Biostrings_2.24.1
[4] GenomicRanges_1.8.11 IRanges_1.14.4       BiocGenerics_0.2.0
```

```
[7] VGAM_0.8-7          aod_1.3

loaded via a namespace (and not attached):
[1] bitops_1.0-4.1 tools_2.15.1   zlibbioc_1.2.0
```