ExomeDepth

Vincent Plagnol

June 26, 2014

Contents

1 What is new?

Version 1.0.5:

- The transition from one CNV state to another now takes into account the distance between exons (contribution from Anna Fowler and Gerton Lunter, Oxford, UK).

- In previous versions, the first exon of each chromosome was set to normal copy number to avoid CNVs spanning multiple exons. This has now been fixed, and the Viterbi algorithm now runs separately for each chromosome (with a dummy exon to start each chromosome that is forced to normal copy number). This should really make a difference for targeted sequencing sets (thank you again to Anna Fowler and Gerton Lunter for suggesting this fix).

- Various small bug fixes, especially to use the `subset.for.speed` argument when a new ExomeDepth object is created.

- Only include primary alignments in the read count function: thank you to Peng Zhang (John Hopkins) for pointing this out.

Version 1.0.0:

- New warning when the correlation between the test and reference samples is low, so that the user knows he should expect unreliable results.
- New function to count everted reads (characteristic of segmental duplications)
- Updated `genes.hg19` and `exons.hg19` objects. The number of exons tested is now 10% lower, because it excludes some non coding regions as well as UTRs that are typically not well covered. My test data suggest that removing these difficult regions cleans up the signal quite a bit and the quality of established CNVs calls has improved significantly as a consequence of this.

2 What ExomeDepth does and tips for QC

2.1 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.

2.2 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.

3 Create count data from BAM files

3.1 Count for the autosomes

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 35138 35174   FAM138A_3
2          1 35277 35481   FAM138A_2
3          1 35721 35736   FAM138A_1
```

```
4          1 53049 53067 AL627309.2_1
5          1 54830 54936 AL627309.2_2
6          1 69091 70008      OR4F5_1
```

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,
+                           include.chr = FALSE,
+                           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). `include.chr` defaults to false: if the BAM file are aligned to a reference sequence with the convention `chr1` for chromosomes instead of simply `1` (i.e. the UCSC convention vs. the Ensembl one) you need to set `include.chr = TRUE`, otherwise the counts will be equal to 0. Note that the data frame with exon locations provided with `ExomeDepth` uses the Ensembl location (i.e. no `chr` prefix before the chromosome names) but what matters to set this option is how the BAM files were aligned.

`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
```

3.2 Counts for chromosome X

Calling CNVs on the X chromosome can create issues if the exome sample of interest and the reference exome samples it is being compared to (what I call the aggregate reference) are not gender matched. For this reason the chromosome X exonic regions are not included by default in the data frame `exons.hg19`, mostly to avoid users getting low quality CNV calls because of this issue. However, loading the same dataset in R also brings another object called `exons.hg19.X` that containts the chromosome X exons.

```
> data(exons.hg19.X)
> head(exons.hg19.X)
```

```
       chromosome  start    end     name
185131          X 200855 200981 PLCXD1_2
185132          X 205400 205536 PLCXD1_3
185133          X 207315 207443 PLCXD1_4
185134          X 208166 208321 PLCXD1_5
185135          X 209702 209885 PLCXD1_6
185136          X 215764 216002 PLCXD1_7
```

This object can be used to generate CNV counts and further down the line CNV calls, in the same way as `exons.hg19`. While this is not really necessary, I would recommend calling CNV on the X separately from the autosomes. Also make sure that the genders are matched properly (i.e. do not use male as a reference for female samples and vice versa).

4 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))
> show(test)
```

```
Number of data points:  26547
Formula:  cbind(test, reference) ~ 1
Phi parameter (range if multiple values have been set):  0.0229405 0.0229405
Likelihood computed
```

5 Build the most appropriate reference set

A key idea behing ExomeDepth is that each exome should not be compared to all other exomes but rather to an optimized set of exomes that are well correlated with that exome. This is what I call the optimized aggregate reference set, which is optimized for each exome sample. So 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)
> print(my.choice[[1]])
```

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

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

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

Note that the `drop = FALSE` option is just used in case the reference set contains a single sample. If this is the case, it makes sure that the subsetted object is a data frame, not a numeric vector.

6 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')
```

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

  start.p end.p        type nexons    start      end chromosome
1      25    27    deletion      3    89553    91106          1
2      62    66    deletion      5   461753   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      chr1:461753-523834  9.82            363            190       0.523
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
```

Now the last thing to do is to save it in an easily readable format (csv in this example, which can be opened in excel if needed):

```
> output.file <- 'exome_calls.csv'
> write.csv(file = output.file,
+           x = all.exons@CNV.calls,
+           row.names = FALSE)
```

Note that it is probably best to annotate the calls before creating that csv file (see below for annotation tools).

7 Ranking the CNV calls by confidence level

ExomeDepth tries to be quite aggressive to call CNVs. Therefore the number of calls may be relatively high compared to other tools that try to accomplish the same task. One important information is the BF column, which stands for Bayes factor. It quantifies the statistical support for each CNV. It is in fact the log10 of the likelihood ratio of data for the CNV call divided by the null (normal copy number). The higher that number, the more confident once can be about the presence of a CNV. While it is difficult to give an ideal threshold, and for short exons the Bayes Factor are bound to be unconvincing, the most obvious large calls should be easily flagged by ranking them according to this quantity.

```
> head(all.exons@CNV.calls[ order ( all.exons@CNV.calls$BF, decreasing = TRUE),])

   start.p end.p        type nexons     start       end chromosome
14    6813  6816    deletion      4  40229386  40240129          1
19   14847 14872    deletion     26 146219129 146244718          1
```

```
12    4451  4461 duplication     11  25599041  25655628          1
11    4263  4267    deletion      5  24287932  24301561          1
21   23032 23039    deletion      8 207718655 207726325          1
20   15186 15194    deletion      9 147850202 147909057          1
                        id   BF reads.expected reads.observed reads.ratio
14   chr1:40229386-40240129 30.4            393            102      0.2600
19 chr1:146219129-146244718 29.4            192             28      0.1460
12   chr1:25599041-25655628 27.3            463            914      1.9700
11   chr1:24287932-24301561 20.1            104              0      0.0000
21 chr1:207718655-207726325 17.0            184             74      0.4020
20 chr1:147850202-147909057 15.7             93              2      0.0215
```

8 Better annotation of CNV calls

Much can be done to annotate CNV calls and this is an open problem. While this is a work in progress, I have started adding basic options. Importantly, the key function uses the more recent syntax from the package `GenomicRanges`. Hence the function will only return a warning and not add the annotations if you use a version of `GenomicRanges` prior to `1.8.10`. The best way to upgrade is probably to use `R 2.15.0` or later and let `Bioconductor` scripts do the install for you. If you use `R 2.14` or earlier the annotation steps described below will probably only return a warning and not annotate the calls.

Perhaps the most useful step is to identify the overlap with a set of common CNVs identified in Conrad et al, Nature 2010. If one is looking for rare CNVs, these should probably be filtered out. The first step is to load these reference data from Conrad et al. To make things as practical as possible, these data are now available as part of ExomeDepth.

```
> data(Conrad.hg19)
> head(Conrad.hg19.common.CNVs)
```

```
GRanges with 6 ranges and 1 metadata column:
      seqnames          ranges strand |    names
         <Rle>       <IRanges>  <Rle> | <factor>
  [1]        1 [10499,  91591]      * |  CNVR1.1
  [2]        1 [10499, 177368]      * |  CNVR1.2
  [3]        1 [82705,  92162]      * |  CNVR1.5
  [4]        1 [85841,  91967]      * |  CNVR1.4
  [5]        1 [87433,  89163]      * |  CNVR1.6
  [6]        1 [87446, 109121]      * |  CNVR1.7
  ---
  seqlengths:
    1 10 11 12 13 14 15 16 17 18 19  2 20 21 22  3  4  5  6  7  8  9  X
   NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
```

Then one can use this information to annotate our CNV calls with the function `AnnotateExtra`.

```
> all.exons <- AnnotateExtra(x = all.exons,
+                            reference.annotation = Conrad.hg19.common.CNVs,
+                            min.overlap = 0.5,
+                            column.name = 'Conrad.hg19')
```

The `min.overlap` argument set to 0.5 requires that the Conrad reference call overlaps at least 50% of our CNV calls to declare an overlap. The `column.name` argument simply defines the name of the column that will store the overlap information. The outcome of this procedure can be checked with:

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

```
  start.p end.p        type nexons    start      end chromosome
1      25    27    deletion      3    89553    91106          1
2      62    66    deletion      5   461753   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     chr1:461753-523834  9.82            363            190       0.523
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
                              Conrad.hg19
1 CNVR1.1,CNVR1.2,CNVR1.5,CNVR1.4,CNVR1.7
2                                    <NA>
3                                    <NA>
4                                CNVR17.1
5                                CNVR17.1
6              CNVR72.3,CNVR72.4,CNVR72.2
```

I have processed the Conrad et al data in the GRanges format. Potentially any other reference dataset could be converted as well. See for example the exon information:

```
> exons.hg19.GRanges <- GRanges(seqnames = exons.hg19$chromosome,
+                                 IRanges(start=exons.hg19$start,end=exons.hg19$end),
+                                 names = exons.hg19$name)
> all.exons <- AnnotateExtra(x = all.exons,
+                            reference.annotation = exons.hg19.GRanges,
+                            min.overlap = 0.0001,
+                            column.name = 'exons.hg19')
> all.exons@CNV.calls[3:6,]
```

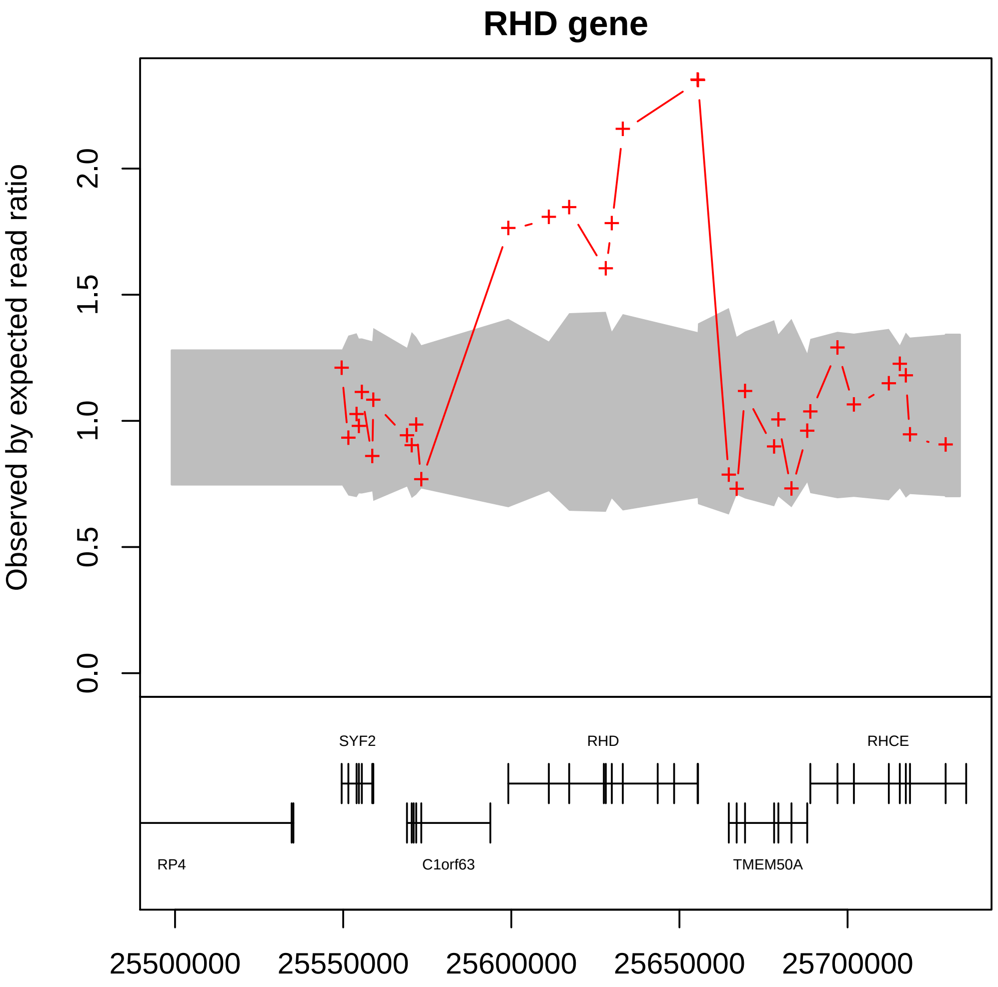
```
  start.p end.p        type nexons    start      end chromosome
3     100   103 duplication      4   743956   745551          1
4     575   576    deletion      2  1569583  1570002          1
5     587   591    deletion      5  1592941  1603069          1
6    2324  2327    deletion      4 12976452 12980570          1
                      id    BF reads.expected reads.observed reads.ratio
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
                 Conrad.hg19
3                       <NA>
4                   CNVR17.1
5                   CNVR17.1
6 CNVR72.3,CNVR72.4,CNVR72.2
                                               exons.hg19
3                                                    <NA>
4                                       MMP23B_7,MMP23B_8
5 SLC35E2B_10,SLC35E2B_9,SLC35E2B_8,SLC35E2B_7,SLC35E2B_6
6                           PRAMEF7_2,PRAMEF7_3,PRAMEF7_4
```

This time I report any overlap with an exon by specifying a value close to 0 in the `min.overlap` argument. Note the metadata column `names` which MUST be specified for the annotation to work properly.

9 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)
```



10 How to loop over the multiple samples

A FAQ is a way to deal with a set of a significant number of exomes, i.e. how to loop over all of them using ExomeDepth. This can be done with a loop. I show below an example of how I would code things. The code is not executed in the vignette to save time when building the package, but it can give some hints to users who do not have extensive experience with R.

```
> #### get the annotation datasets to be used later
> data(Conrad.hg19)
> exons.hg19.GRanges <- GRanges(seqnames = exons.hg19$chromosome,
+                               IRanges(start=exons.hg19$start,end=exons.hg19$end),
+                               names = exons.hg19$name)
> ### prepare the main matrix of read count data
> ExomeCount.mat <- as.matrix(ExomeCount.dafr[, grep(names(ExomeCount.dafr), pattern = 'Exome.*')])
> nsamples <- ncol(ExomeCount.mat)
> ### start looping over each sample
> for (i in 1:nsamples) {
+
+ #### Create the aggregate reference set for this sample
+   my.choice <- select.reference.set (test.counts =  ExomeCount.mat[,i],
+                                      reference.counts = ExomeCount.mat[,-i],
+                                      bin.length = (ExomeCount.dafr$end - ExomeCount.dafr$start)/1000,
+                                      n.bins.reduced = 10000)
+
+   my.reference.selected <- apply(X = ExomeCount.mat[, my.choice$reference.choice, drop = FALSE],
+                                  MAR = 1,
+                                  FUN = sum)
+
+   message('Now creating the ExomeDepth object')
+   all.exons <- new('ExomeDepth',
+                    test = ExomeCount.mat[,i],
+                    reference = my.reference.selected,
+                    formula = 'cbind(test, reference) ~ 1')
+
+ ################ Now call the CNVs
+   all.exons <- CallCNVs(x = all.exons,
+                         transition.probability = 10^-4,
+                         chromosome = ExomeCount.dafr$space,
+                         start = ExomeCount.dafr$start,
+                         end = ExomeCount.dafr$end,
+                         name = ExomeCount.dafr$names)
+
+ ########################### Now annotate the ExomeDepth object
+   all.exons <- AnnotateExtra(x = all.exons,
+                              reference.annotation = Conrad.hg19.common.CNVs,
+                              min.overlap = 0.5,
+                              column.name = 'Conrad.hg19')
+
+   all.exons <- AnnotateExtra(x = all.exons,
+                              reference.annotation = exons.hg19.GRanges,
+                              min.overlap = 0.0001,
+                              column.name = 'exons.hg19')
+
+   output.file <- paste('Exome_', i, 'csv', sep = '')
+   write.csv(file = output.file, x = all.exons@CNV.calls, row.names = FALSE)
+
+ }
>
>
```

11 Additional functions

11.1 Matched tumour/normal tissue pair

A few users asked me about a typical design in the cancer field of combining sequence data for healthy and tumour tissue. A function has been available for this for a long time: `somatic.CNV.call`. It replaces the test sample with the tumour sample, and the reference sample with the healthy tissue. It also requires as an input a parameter that specifies the proportion of the tumour sample that is also from the tumour. This function is a bit experimental but I welcome feedback.

11.2 Counting everted reads

This issue of everted reads (read pairs pointing outward instead of inward) is a side story that has nothing to do with read depth. It is based on a "read pair" analysis but it was practical for me (and I hope for others) to write simple functions to find such a pattern in my exome data. The motivation is that everted reads are characteristic of tandem duplications, which could be otherwised by ExomeDepth. I have argued before that such a paired end approach is not well suited for exome sequence data, because the reads must be located very close to the junction of interest. This is not very likely to happen if one only sequences 1% of the genome (i.e. the exome). However, experience shows that these reads can sometimes be found, and it can be effective at identifying duplications. The function below will return a data frame with the number of everted reads in each gene. More documentation is needed on this, and will be added in later versions. But it is a useful additional screen to pick up segmental duplications.

```
> data(genes.hg19)
> everted <- count.everted.reads (bed.frame = genes.hg19,
+                                 bam.files = bam.files.list,
+                                 min.mapq = 20,
+                                 include.chr = FALSE)
>
```

Note the option `include.chr = FALSE` is you aligned your reads to the NCBI version of the reference genome (i.e. chromosome denoted as 1,2, ...) but set to `TRUE` if you used the UCSC reference genome (i.e. chromosomes denoted as chr1, chr2, ...). As a filtering step one probably wants to look in the resulting table for genes that show no everted reads in the vast majority of samples, with the exception of a few cases where a duplication may be rare but real.

12 Technical information about R session

```
> sessionInfo()

R version 3.0.2 (2013-09-25)
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=en_US.iso885915       LC_NAME=C
 [9] LC_ADDRESS=C                   LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.iso885915 LC_IDENTIFICATION=C

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

other attached packages:
[1] ExomeDepth_1.0.5     Rsamtools_1.14.3     Biostrings_2.30.1
[4] GenomicRanges_1.14.4 XVector_0.2.0        IRanges_1.20.7
```

```
[7] BiocGenerics_0.8.0   VGAM_0.9-4           aod_1.3

loaded via a namespace (and not attached):
[1] bitops_1.0-6   tools_3.0.2    zlibbioc_1.8.0
```