Introduction to GeneBreak

Evert van den Broek[*]& Stef van Lieshout

July 14, 2015

Department of Pathology
VU University Medical Center
The Netherlands, Amsterdam

Contents

[*] email@email.com

1 Running GeneBreak

This is a short tutorial on how to use the GeneBreak package. It describes an example workflow which uses included copy number data of one chromosome from 200 aCGH samples. Let's start by loading the package.

```
> library(GeneBreak)
```

1.1 Detect breakpoints from copy-number data

Copy number data can be loaded in two ways. Either from a cghCall/QDNAseq object (ouput of bioconductor packages CGHcall or QDNAseq) or by providing a data.frame with at least 5 columns: Chromosome, Start, End and FeatureName (usually probe identifier). Note: when using the data.frame input the column names must be exactly as described here and in the same order! In this tutorial we will use a built-in dataset of chromosome 20:

1.1.1 Loading cghCall object

```
> data( "copynumber.data.chr20" )
```

By inspecting the dataset, we see that we are dealing with an R object of class "cghCall" with 3653 features (aCGH probes in this case) and 200 samples.

```
> copynumber.data.chr20

cghCall (storageMode: lockedEnvironment)
assayData: 3653 features, 200 samples
  element names: calls, copynumber, probamp, probgain, probloss, probnorm, segmented
protocolData: none
phenoData
  sampleNames: sample_1 sample_2
    ... sample_200 (200 total)
  varLabels: Cellularity
  varMetadata: labelDescription
featureData
  featureNames: A_16_P03469195
    A_14_P136138 ... A_18_P13856091
    (3653 total)
  fvarLabels: Chromosome Start End
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:

> breakpoints <- getBreakpoints( data = copynumber.data.chr20 )

Breakpoint detection started...
```

1.1.2 Loading data from data.frame()

Although we recommend the usage of either `QDNAseq` (for sequencing data) or `CGHcall` (for aCGH data) as a precursor for GeneBreak, there is a possibilty of using a `data.frame()` as input. This allows for the analysis of data from any pipeline by importing a text file into R.

Here we use the output of CGHcall to output two data.frames() with segment and (optionally) call values. These can be used as input instead of a cghCall or QDNAseq object in `getBreakpoints`.

```
> library(CGHcall)
> cgh <- copynumber.data.chr20
> segmented <- data.frame( Chromosome=chromosomes(cgh), Start=bpstart(cgh),
+  End=bpend(cgh), FeatureName=featureNames(cgh), segmented(cgh))
> called <- data.frame( Chromosome=chromosomes(cgh), Start=bpstart(cgh),
+  End=bpend(cgh), FeatureName=featureNames(cgh), calls(cgh))
> breakpoints <- getBreakpoints( data = segmented, data2 = called )
```

1.2 Loading gene annotation data

Then we need to obtain gene annotations. For hg18 (and hg19, hg38) reference sequence these are included and can be loaded:

```
> data( "ens.gene.ann.hg18" )
```

Inspect the annotation.

```
> head( ens.gene.ann.hg18 )
      Gene           EnsID Chromosome  Start    End   band strand
MIRN1302-2 ENSG00000221311          1  20229  20366 p36.33      1
   FAM138E ENSG00000222027          1  24417  25944 p36.33     -1
   FAM138E ENSG00000222003          1  24417  25944 p36.33     -1
   FAM138A ENSG00000222003          1  24417  25944 p36.33     -1
     OR4F5 ENSG00000177693          1  58954  59871 p36.33      1
    OR4F29 ENSG00000177799          1 357522 358460 p36.33      1
```

1.3 Breakpoint selection by filtering

Next we filter breakpoints. Different filters can be set with different threshold. Default here is ”deltaSeg” filter with a threshold of 0.2. This means that only breakpoints which...

```
> breakpointsFiltered <- bpFilter( breakpoints )
```

```
Applying BP selection...
```

Next we will add the gene annotation information to the GeneBreak object. No analysis is done here yet.

1.4 Detection of gene associated breaks

```
> breakpointsAnnotated <- addGeneAnnotation( breakpointsFiltered, ens.gene.ann.hg18 )

Adding of gene annotation started on 659 genes by 200 samples
0% ... 25% ... 50% ... 75% ... Adding gene annotation DONE
```

Next we perform the gene analysis. This overlaps the genomic locations of the genes with the copy number data to find breakpoints within genes.

```
> breakpointGenes <- bpGenes( breakpointsAnnotated )

Running bpGenes: 659 genes and 200 samples
0% ... 25% ... 50% ... 75% ... bpGenes DONE
A total of 1029 gene breaks in 241 genes detected
```

Next we determine the significantly recurring breakpoints. This be done at the ”gene” or ”feature” level and using one of two different methods (”Benjamini Hochberg” or ”Gilbert”). The advantage of using... NOTE: when running bpStats() many warnings can be generated by a function (glm.fit) of a dependancy package, this does not harm the analysis.

```
> breakpointStatistics <- bpStats( breakpointGenes )

Applying statistical test over 200 samples for: gene breakpoints: BH test...

> breakpointStatistics
```

This will return an object of class `CopyNumberBreakPointGenes`.

By using recurrentGenes() we can observe the recurrent affected genes.

```
> head( recurrentGenes( breakpointStatistics ) )

 A total of 14 recurrent breakpoint genes (at FDR < 0.1)
          Gene sampleCount featureTotal
13886   PCMTD2          64            4
13898 C20orf69          33            3
4268     BFSP1           8            5
5473    ABHD12          10            9
4780  C20orf26           7           18
3493      HAO1           5            5
             pvalue          FDR
13886 1.350385e-103 8.899035e-101
13898  5.522293e-44  1.819595e-41
4268   3.941447e-07  8.658045e-05
5473   5.756361e-05  9.483605e-03
4780   2.748743e-04  3.622843e-02
3493   6.528175e-04  3.961727e-02
```

1.5 Plotting the frequencies of breakpoint locations

```
> bpPlot( breakpointStatistics )

Plotting breakpoint frequencies ...
Plotting Chromosome: 20
```

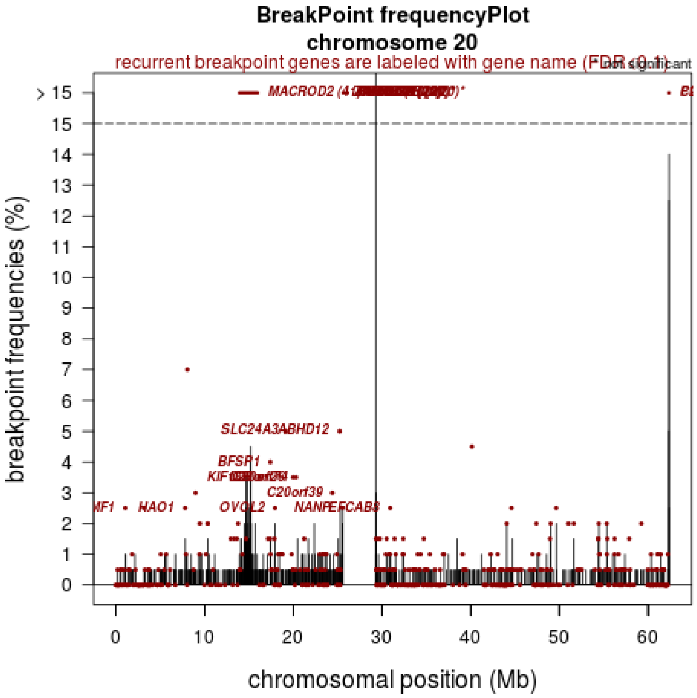

Figure 1: Caption

2 Storage of R objects

At any time during the analysis, the GeneBreak objects (and any R objects for that matter) can be saved to disk with: `saveRDS`, and in the future be read from the local file with `loadRDS`

3 Downloading Gene Annotations

This section describes the steps taken to create the gene annotations used in this package. It may serve as a start for creating your own if required for whatever reason.

```
> # gene annotations obtained via Biomart.
> # HUGO gene names (HGNC symbol), Ensembl_ID and chromosomal location
>
> # Used (and most) recent releases:
> # HG18: release54
> # HG19: release75
> # HG38: release80 (date: 150629)
>
> library(biomaRt)
> ensembl54 = useMart( host='may2009.archive.ensembl.org', biomart='ENSEMBL_MART_ENSEMBL', d
> ensembl75 = useMart( host='feb2014.archive.ensembl.org', biomart='ENSEMBL_MART_ENSEMBL', d
> ensembl80 = useMart( "ensembl", dataset="hsapiens_gene_ensembl" )
> createAnnotationFile <- function( biomartVersion ) {
+   biomart_result <- getBM(attributes =  c("hgnc_symbol", "ensembl_gene_id", "chromosome_na
+
+   biomart_result[ ,3] <- as.vector( biomart_result[ ,3] )
+   biomart_result$chromosome_name[ biomart_result$chromosome_name=="X" ] <- "23"
+   biomart_result$chromosome_name[ biomart_result$chromosome_name=="Y" ] <- "24"
+
+   biomart_genes <-biomart_result[ which(biomart_result[ ,1]!="" & biomart_result[ ,3] %in%
+   colnames(biomart_genes)[1:5]<-c("Gene","EnsID","Chromosome","Start","End")
+
+   cat( c("Biomart version:", biomartVersion@host, "including:", dim(biomart_genes)[1], "ge
+   return( biomart_genes )
+ }
> ens.gene.ann.hg18 <- createAnnotationFile( ensembl54 )
> ens.gene.ann.hg19 <- createAnnotationFile( ensembl75 )
> ens.gene.ann.hg38 <- createAnnotationFile( ensembl80 )
>
```

4 Session Information

The version number of R and packages loaded for generating the vignette were:

```
R version 3.2.1 (2015-06-18)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 14.04.2 LTS

locale:
 [1] LC_CTYPE=en_US.UTF-8
 [2] LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8
 [6] LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8
 [8] LC_NAME=C
 [9] LC_ADDRESS=C
[10] LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8
[12] LC_IDENTIFICATION=C

attached base packages:
[1] parallel  stats     graphics
[4] grDevices utils     datasets
[7] methods   base

other attached packages:
[1] CGHbase_1.26.0
[2] marray_1.44.0
[3] limma_3.22.7
[4] Biobase_2.26.0
[5] BiocGenerics_0.12.1
[6] GeneBreak_0.99.0

loaded via a namespace (and not attached):
[1] tools_3.2.1
```