How to use breakpointR

David Porubsky*

April 30, 2018

Contents

*david.porubsky@gmail.com

1 Introduction

BreakpointR is a novel algorithm designed to accurately tracks template strand changes in Strand-seq data using a bi-directional read-based binning. Read-based binning strategy scales bin size dynamically to accommodate defined number of reads what accounts for mappability bias in sparsely covered single-cell Strand-seq data. In such dynamically scaled bins, read directionality is tracked in order to search for points where template strand state changes. BreakpointR takes as an input reads aligned to the reference genome stored in BAM files. BreakpointR outputs locations where directionality of sequenced teplate strands changes.

2 Quickstart

The main function of this package is called `breakpointr()` and performs all the necessary steps to get from aligned reads in BAMs to interpretable output. For an unexperienced user we advise to run *breakpointR* with default parameters and later based on the obtained results start to tweak certain parameters. For more datailed guidance on parameter tweaking see section 3.

```
library(breakpointR)
# Run breakpointR with default paprameters
breakpointr(inputfolder='folder-with-BAM', outputfolder='output-directory')
```

Although in most cases the one of the above commands will produce reasonably good results, it might be worthwile to adjust the default parameters to improve performance and the quality of the results. You can get a description of all available parameters by typing

```
?breakpointr
```

After the function has finished, you will find the folder **output-directory** containing all produced files and plots. This folder contains the following *files* and **folders**:

- *breakpointR.config*: This file contains all the parameters that are necessary to reproduce your analysis. You can specify this file as

  ```
  breakpointr(..., configfile='breakpointR.config')
  ```

  to run another analysis with the same parameter settings.
- **breakpoints** UCSC browser formated bedgraphs compiling all breakpoints across all single-cell libraries. This folder also contains list of all localized breakpoints in all single-cell libraries. Lastly, locations of breakpoint hotspots are reported here if

  ```
  callHotSpots=TRUE
  ```
- **browserfiles** UCSC browser formated files with exported reads, deltaWs and breakPoints for every single-cell library.
- **data** Contains RData files storing results of BreakpointR analysis for each single-cell library.
- **plots**: Genome-wide plots for selected chromsosome, genome-wide heatmap of strand states as well as chromosome specific read distribution together with localized breakpoints. All plots are created by default.

  ```
  #Eventually plots can be created from stored results.
  files2plot <- list.files('output-directory/data/', full.names=TRUE)
  breakpoint.plots <- plotBreakpoints(files2plot)
  breakpointPerChr.plots <- plotBreakpointsPerChr(files2plot)
  genome.heatmap <- plotHeatmap(files2plot)
  ```

3 Recommended settings

3.1 Reading BAM files

Currently *breakpointR* can take as an input only aligned reads stored in BAM files. All BAM files are expected to be present in a folder specified in `breakpointr(..., inputfolder)`. We advise to remove reads with low mapping quality. Duplicated reads are always removed.

```
breakpointr(..., min.mapq=10)
```

3.2 Removing certain regions

breakpointR allows a user to exclude certain genomic regions from the analysis. This comes handy when one wants to remove reads that falls into low complexity regions such as segmental duplications or centromeres. To mask certain genomic regions user has to define option `breakpointr(..., maskRegions)` to a bed formated text file. All reads falling into these regions will be discarded prior to breakpoint detection. User defined regions to mask can be downloaded from the UCSC Table Browser.

3.3 Binning strategy

breakpointR uses read based binning strategy and offers two approaches to set the bin size: (1) user defined number of reads in each bin or (2) number of reads in every bin is selected based on desired bin length.

```
library(breakpointR)
# Binning strategy based on desired bin length
breakpointr(inputfolder='folder-with-BAM', outputfolder='output-directory', windowsize=1e6,
            binMethod='size')
# Binning strategy based user-defined number of reads in each bin
breakpointr(inputfolder='folder-with-BAM', outputfolder='output-directory', windowsize=100,
            binMethod='reads')
```

Based on the size of the user defined bin changes sensitivity and specifity of breakpoint detection. We recomend to select rather large bin size (>=1Mb) in order to reliably detect low frequency SCE events. In order to detect smaller events like inversions smaller bin size is recommended. Keep in mind that such settings also leads to higher level of false positive breakpoints. In this case one might need to tweak some breakpoint detection parameters (see subsection 3.4).

3.4 Breakpoint peak detection

Breakpoint detection is based on finding significant peaks in deltaW values. Level of signifcance is measured in z-score (or number SDs) from the set threshold `breakpointr(..., peakTh)`. By default the threshold is set to the 1/3 of the highest detlaW value. For the data with noisy coverage we recommend to set this threshold little bit higher, for example 1/2 of the highest deltaW value. In case of noisy data we also recommend to tweak trim option `breakpointr(..., trim)` which used to calculate SD after trimming extreme deltaW values.

```
# Example deltaW values
exampleFolder <- system.file("extdata", "example_results", package="strandseqExampleData")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
breakpoint.object <- loadFromFiles(exampleFile)
breakpoint.object[[1]]$deltas
```

```
## GRanges object with 829044 ranges and 1 metadata column:
##            seqnames                ranges strand |    deltaW
##               <Rle>             <IRanges>  <Rle> | <numeric>
```

```
##         [1]     chr1         [ 7560,  7594]      - |        17
##         [2]     chr1         [ 8569,  8612]      + |         0
##         [3]     chr1         [15116, 15143]      - |        57
##         [4]     chr1         [17235, 17240]      - |       130
##         [5]     chr1         [19615, 19720]      - |        41
##         ...      ...                    ...    ... .       ...
##    [829040]     chrX [155995858, 155997941]      - |         0
##    [829041]     chrX [155998301, 155999275]      - |         0
##    [829042]     chrX [155999653, 156003560]      + |         0
##    [829043]     chrX [156004027, 156030352]      + |         0
##    [829044]     chrX [156030727, 156040895]      - |         0
##    -------
##    seqinfo: 23 sequences from an unspecified genome
```

3.5 Background reads

Backround reads are a common feature of Strand-seq libraries. Strand-seq is based on removal of newly synthesized strand during DNA replication, however this process is not perfect. Therefore, we usually expect low abundance reads aligned in opposite direction even for puraly WW or CC chromosomes. Another reason to see such artefacs is imperfect read mapping especially in repetitive regions. To remove reads falling into the repetitive regions see subsection 3.2.

3.6 Calling breakpoint hotspots

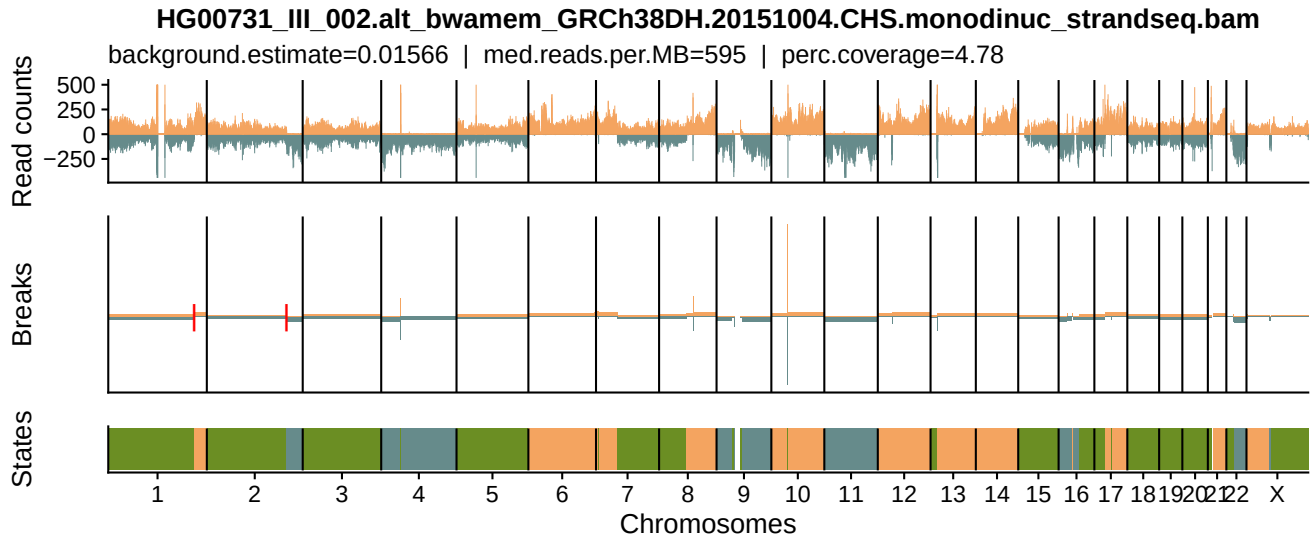
In order to find locations where breakpoints occur around the same genomic position in multiple Strand-seq libraries there is `hotspotter()`. Function can be invoked by setting corresponding parameter to 'TRUE'. It make sense to set this parameter onyl if you have available reasonable number (>=50) of Strand-seq libraries.

```
breakpointr(..., callHotSpots=TRUE)
```

3.7 Loading results and plotting single cells

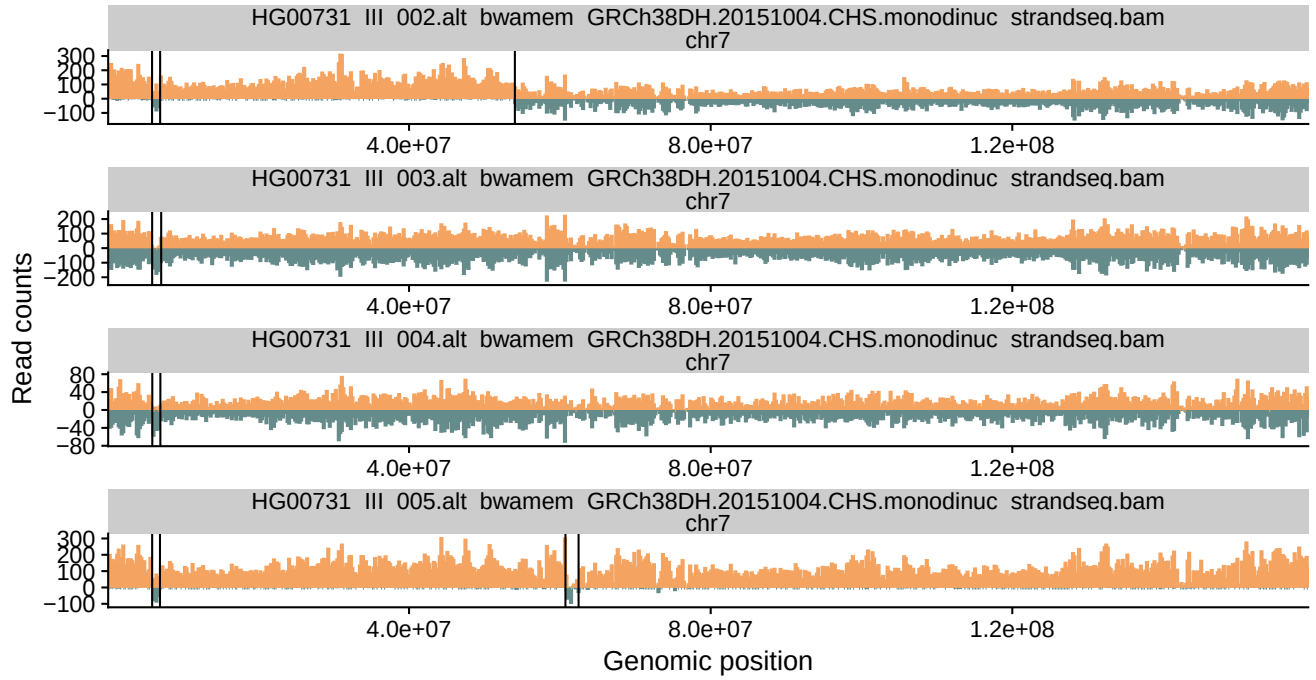
```
# Plotting a single library
exampleFolder <- system.file("extdata", "example_results", package="strandseqExampleData")
exampleFile <- list.files(exampleFolder, full.names=TRUE)[1]
plotBreakpoints(exampleFile)
```

```
## [[1]]
```




```
# Plotting a single library
exampleFolder <- system.file("extdata", "example_results", package="strandseqExampleData")
exampleFiles <- list.files(exampleFolder, full.names=TRUE)[1:4]
plotBreakpointsPerChr(exampleFiles, chromosomes = 'chr7')

## $chr7
```



4 Session Info

```
toLatex(sessionInfo())
```

- R version 3.4.4 (2018-03-15), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=en_US.UTF-8, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Running under: Debian GNU/Linux 9 (stretch)
- Matrix products: default
- BLAS: /usr/lib/atlas-base/atlas/libblas.so.3.0
- LAPACK: /usr/lib/atlas-base/atlas/liblapack.so.3.0
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.20.0, breakpointR 1.0.0, cowplot 0.9.2, GenomeInfoDb 1.10.3, GenomicRanges 1.26.4, ggplot2 2.2.1, IRanges 2.8.2, knitr 1.17, S4Vectors 0.12.2, strandseqExampleData 0.99.0
- Loaded via a namespace (and not attached): Biobase 2.34.0, BiocParallel 1.8.2, BiocStyle 2.2.1, Biostrings 2.42.1, bitops 1.0-6, codetools 0.2-15, colorspace 1.3-2, compiler 3.4.4, doParallel 1.0.11, evaluate 0.10.1, foreach 1.4.4, GenomicAlignments 1.10.1, grid 3.4.4, gtable 0.2.0, gtools 3.5.0, highr 0.6, iterators 1.0.9, labeling 0.3, lattice 0.20-35, lazyeval 0.2.1, magrittr 1.5, Matrix 1.2-12, munsell 0.4.3, pillar 1.2.1, plyr 1.8.4, Rcpp 0.12.16, RCurl 1.95-4.8, rlang 0.2.0, Rsamtools 1.26.2, scales 0.5.0, stringi 1.1.7, stringr 1.3.0, SummarizedExperiment 1.4.0, tibble 1.4.2, tools 3.4.4, XVector 0.14.1, yaml 2.1.18, zlibbioc 1.20.0