EMBL single-cell Omics Course

David Porubsky

May 5, 2019

Contents

1 Introduction

In this practical we will learn how to go from raw Strand-seq reads, through mapping strand-state breakpoints and we will finish by phasing of haplotype informative Strand-seq reads.

2 Mapping strand-state changes using breakpointR

The main function of this package is called `breakpointr()` and performs all the necessary steps to get from aligned reads in BAMs to a set of breakpoints that mark strand-state changes.

```
## Install breakpointR package from Bioconductor
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("breakpointR")

## Load breakpointR package
library(breakpointR)

## Run breakpointR with default parameters
breakpointr(inputfolder='folder-with-BAMs', outputfolder='output-folder')
```

Although in most cases the one of the above commands will produce reasonably good results, however, in this course we will learn more about various parameters settings used in breakpointR and how it used in various scenarios.

```
## To get a help page explaining breakpointR parameter settings
?breakpointr
```

In this practical we will work in this predefined directory

```
## Set working directory
setwd("/home/porubsky/WORK/EMBL_Omics_practical/")
```

2.1 Running breakpointR

The function `breakpointr()` takes an input BAM files stored in the inputfolder and produces an outputfolder with results, plots and browserfiles. The following code is an example of how to run *breakpointR* for single-end reads with windowsize defined by size.

To run breakpointR using our example data type/copy the code below into your Rstudio session.

```
## Run breakpointR
breakpointr(inputfolder = "bams/",
            outputfolder = "BreakpointR_results",
            windowsize = 2000000,
            binMethod = 'size',
            pairedEndReads = TRUE,
```

```
                chromosomes = 'chr16',
                min.mapq = 10,
                filtAlt = TRUE)
```

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.
- **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 chromosomes, genome-wide heatmap of strand states as well as chromosome specific read distribution together with localized breakpoints. All aformentioned plots are created by default.

Results will be stored in **outputfolder/data** as RData objects. Such data can be later loaded for further processing and customized plotting.

To load breakpointR results into the R.

```
## Load results for one Strand-seq library
files <- list.files(path = "BreakpointR_results/data/",
                    pattern = "RData",
                    full.names = TRUE)

tmp <- get(load(files[1]))
tmp
```

```
## $ID
## [1] "HG00733_I_002.alt_bwamem_GRCh38DH.20151004.CHS.monodinuc_strandseq.bam"
##
## $fragments
## GRanges object with 45542 ranges and 1 metadata column:
##           seqnames            ranges strand |      mapq
##              <Rle>         <IRanges>  <Rle> | <integer>
##       [1]    chr16       18923-19073      + |        45
##       [2]    chr16       35998-36148      - |        60
##       [3]    chr16       37318-37393      - |        60
##       [4]    chr16       38079-38154      - |        60
##       [5]    chr16       39413-39488      - |        60
##       ...      ...               ...    ... .       ...
##   [45538]    chr16 90079738-90079813      - |        60
```

```
##   [45539]    chr16 90080096-90080171      - |        60
##   [45540]    chr16 90169006-90169081      - |        21
##   [45541]    chr16 90169320-90169470      - |        27
##   [45542]    chr16 90214955-90215030      - |        22
##   -------
##   seqinfo: 1 sequence from an unspecified genome
##
## $deltas
## GRanges object with 32510 ranges and 1 metadata column:
##           seqnames            ranges strand |    deltaW
##              <Rle>         <IRanges>  <Rle> | <numeric>
##       [1]    chr16       19073-35998      + |         0
##       [2]    chr16       36148-37318      - |         0
##       [3]    chr16       37393-38079      - |         0
##       [4]    chr16       38154-39413      - |         0
##       [5]    chr16       39488-43064      - |         0
##       ...      ...               ...    ... .       ...
##   [32506]    chr16 90079813-90080096      - |         0
##   [32507]    chr16 90080171-90169006      - |         0
##   [32508]    chr16 90169081-90169320      - |         0
##   [32509]    chr16 90169470-90214955      - |         0
##   [32510]    chr16 90215030-90338345      - |         0
##   -------
##   seqinfo: 1 sequence from an unspecified genome
##
## $breaks
## GRanges object with 4 ranges and 2 metadata columns:
##       seqnames            ranges strand |       genoT    deltaW
##          <Rle>         <IRanges>  <Rle> | <character> <numeric>
##   [1]    chr16 21320820-21452980      * |       ww-cc       505
##   [2]    chr16 22695453-23086611      * |       cc-ww       499
##   [3]    chr16 34595531-34596046      * |       ww-cc       956
##   [4]    chr16 46400893-46402234      * |       cc-ww      1037
##   -------
##   seqinfo: 1 sequence from an unspecified genome
##
## $confint
## GRanges object with 4 ranges and 2 metadata columns:
##       seqnames            ranges strand |       genoT    deltaW
##          <Rle>         <IRanges>  <Rle> | <character> <numeric>
##   [1]    chr16 21188850-21567520      * |       ww-cc       505
##   [2]    chr16 22690470-23205320      * |       cc-ww       499
##   [3]    chr16 34591965-34738837      * |       ww-cc       956
##   [4]    chr16 46390708-46466054      * |       cc-ww      1037
##   -------
##   seqinfo: 1 sequence from an unspecified genome
##
## $counts
## GRanges object with 5 ranges and 3 metadata columns:
##       seqnames            ranges strand |        Ws        Cs      states
##          <Rle>         <IRanges>  <Rle> | <integer> <integer> <character>
```

```
##   [1]    chr16        1-21320819      * |     10449        52          ww
##   [2]    chr16 21452981-22695452      * |        88       505          cc
##   [3]    chr16 23086612-34595530      * |      5174       108          ww
##   [4]    chr16 34596047-46400892      * |        42      1250          cc
##   [5]    chr16 46402235-90338345      * |     27417       148          ww
##   -------
##   seqinfo: 1 sequence from an unspecified genome
##
## $lib.metrics
## background.estimate    med.reads.per.MB       perc.coverage
##        7.225333e-03        5.810000e+02        3.360000e+00
##
## $params
##     windowsize      binMethod           trim         peakTh           zlim     background
##        "2e+06"         "size"           "10"         "0.33"        "3.291"         "0.05"
##       minReads confidence.int
##           "10"         "0.99"
##
## attr(,"class")
## [1] "BreakPoint"
```

Next we will explore how to load USCS formatted bed files into the UCSC genome browser. As mentioned earlier such files are stored in browserfiles folder inside data analysis directory. Go to https://genome.ucsc.edu/ and under **'Genomes'** select **'Human GRCh38'**. Use **'hide all'** at the bottom of the UCSC screen in order to hide all tracks. In order to load your custom tracks produced by breakpointR go to **'add custom tracks'** and in the next window go to browse and select UCSC formatted files from **BreakpointR_results/browserfiles**.

2.2 Compile all directional reads into a single composite file

However, coverage of every single-cell library is rather low what prevents to map smaller changes in read directionality. therefore in order to increase coverage we will concatenate all reads across multiple Strand-seq libraries while at the same time we will synchronize read directionality across those libraries.

```
## List all files
files <- list.files(file.path("BreakpointR_results/data/"),
                    full.names = T)
syncFrags <- synchronizeReadDir(files2sync = files)
## Run breakpointR on composite data
breakpoints <- runBreakpointr(bamfile = syncFrags,
                              pairedEndReads = FALSE,
                              chromosomes = 'chr16',
                              windowsize = 100000,
                              binMethod = 'size')

## Write out the final results
breakpointr2UCSC(index="chr16_compositeFile",
                 outputDirectory="BreakpointR_results/",
                 fragments=breakpoints$fragments,
                 deltaWs=breakpoints$deltas,
                 breakTrack=breakpoints$breaks,
                 confidenceIntervals=breakpoints$confint)
```

2.3 Report haplotype informative regions

As was presented before, Strand-seq carry haplotype information in every cell for chromosomes that inherited Watson and Crick template. In these cases we can distinguish parental homologs based on directionality of single-stranded DNA.

```
## Export haplotype informative regions (WC state)
exportRegions(datapath = "BreakpointR_results/data/",
              file = "HG00733_WCregions.txt",
              collapseInversions = TRUE,
              collapseRegionSize = 4000000,
              minRegionSize = 5000000,
              state = 'wc')
```

3 Phasing haplotype informative regions using StrandPhaseR

In order to phase these haplotype informative regions we will use another R package called StrandPhaseR.

```
## To install StrandPhaseR
install_github("daewoooo/StrandPhaseR", force=TRUE)
```

3.1 Get set of SNVs for phasing

As explained earlier haplotypes represent set of variants (SNVs) that are linked on a single parental homolog. To phase these variants using StrandPhaseR we need to know their location in the genome. We can get such information from hight coverage data such as bulk Illumina sequencing or linked-read sequencing by 10X genomics.

```
## Filter SNVs using bcftools
## Use -m2 -M2 -types snps to only view biallelic SNPs.
bcftools view -m2 -M2 --types snps --genotype het --apply-filter PASS \
/snvs/HG00733_WGS_phased_variants.vcf.gz \
> HG00733_WGS_phased_variants_filtered.vcf
```

3.2 Perform phasing

Perform phasing on a set of known variable positons (SNVs) over haplotype informative regions in every single cell.

```
## Load StrandPhaseR package
library(StrandPhaseR)

## Run StrandPhaseR
strandPhaseR(inputfolder = "bams/",
             outputfolder = "StrandPhaseR_results",
             positions = "snvs/HG00733_WGS_phased_variants_chr16_filtered.vcf",
             WCregions = "HG00733_WCregions.txt",
             chromosomes = 'chr16',
             pairedEndReads = TRUE,
             min.mapq = 10,
             min.baseq = 20,
             splitPhasedReads = TRUE
             )
```

3.3 Detect structural changes phased haplotypes

Sometimes we want to detect strand-state changes in phased reads. For this we consider haplotype 1 reads as Crick and haplotype 2 reads as Watson. This will allow us to detect heterozygous deletions in our data.

```
## Run breakpointR on phased reads
phased.reads <- list.files("StrandPhaseR_results/data/",
                           pattern = "reads.RData",
                           full.names = TRUE)

## Merge all phased reads
all.reads <- GRangesList()
seqlevels(all.reads) <- 'chr16'
for (i in seq_along(phased.reads)) {
    reads <- get(load(phased.reads[i]))
    strand(reads$hap1) <- "+"
    strand(reads$hap2) <- "-"
    reads <- c(reads$hap1, reads$hap2)
    reads <- sort(reads, ignore.strand=TRUE)
    all.reads[[i]] <- reads
}
merged.reads <- unlist(all.reads)

## Run breakpointR
breakpoints <- runBreakpointr(bamfile = merged.reads,
                              pairedEndReads = FALSE,
                              chromosomes = 'chr16',
                              windowsize = 100000,
                              binMethod = "size",
                              background = 0.1,
                              peakTh = 0.25)

## Write out the final results
breakpointr2UCSC(index="phasedReads_chr16",
                 outputDirectory="StrandPhaseR_results/",
                 fragments=breakpoints$fragments,
                 deltaWs=breakpoints$deltas,
                 breakTrack=breakpoints$breaks,
                 confidenceIntervals=breakpoints$confint)
```

4 Session Info

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

- R version 3.5.3 (2019-03-11), `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: `Ubuntu 18.04.2 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.28.0, breakpointR 1.1.3, breakpointRdata 1.0.0, cowplot 0.9.4, GenomeInfoDb 1.18.2, GenomicRanges 1.34.0, ggplot2 3.1.1, IRanges 2.16.0, knitr 1.22, reshape2 1.4.3, S4Vectors 0.20.1, StrandPhaseR 0.1
- Loaded via a namespace (and not attached): AnnotationDbi 1.44.0, assertthat 0.2.1, Biobase 2.42.0, BiocManager 1.30.4, BiocParallel 1.16.6, BiocStyle 2.10.0, biomaRt 2.38.0, Biostrings 2.50.2, bit 1.1-14, bit64 0.9-7, bitops 1.0-6, blob 1.1.1, BSgenome 1.50.0, codetools 0.2-16, colorspace 1.4-1, compiler 3.5.3, crayon 1.3.4, DBI 1.0.0, DelayedArray 0.8.0, digest 0.6.18, doParallel 1.0.14, dplyr 0.8.0.1, evaluate 0.13, fastseg 1.28.0, foreach 1.4.4, GenomeInfoDbData 1.2.0, GenomicAlignments 1.18.1, GenomicFeatures 1.34.8, glue 1.3.1, grid 3.5.3, gtable 0.3.0, gtools 3.8.1, highr 0.8, hms 0.4.2, htmltools 0.3.6, httr 1.4.0, iterators 1.0.10, lattice 0.20-38, lazyeval 0.2.2, magrittr 1.5, Matrix 1.2-16, matrixStats 0.54.0, memoise 1.1.0, munsell 0.5.0, pillar 1.3.1, pkgconfig 2.0.2, plyr 1.8.4, prettyunits 1.0.2, progress 1.2.0, purrr 0.3.2, R6 2.4.0, Rcpp 1.0.1, RCurl 1.95-4.12, rlang 0.3.4, rmarkdown 1.12, Rsamtools 1.34.1, RSQLite 2.1.1, rtracklayer 1.42.2, scales 1.0.0, stringi 1.4.3, stringr 1.4.0, SummarizedExperiment 1.12.0, tibble 2.1.1, tidyr 0.8.3, tidyselect 0.2.5, tinytex 0.12, tools 3.5.3, VariantAnnotation 1.28.13, withr 2.1.2, xfun 0.6, XML 3.98-1.19, XVector 0.22.0, yaml 2.2.0, zlibbioc 1.28.0, zoo 1.8-5