EMBL predoc course 2019 Day3 BreakpointR

David Porubsky, modified by Hyobin Jeong

October 18, 2019

Contents

1 Introduction 2
2 Preparation of input files for breakpointR 2
3 Mapping strand-state changes using breakpointR 2
4 Running breakpointR. 3
5 Session Info 5

1 Introduction

In this practical we will learn how to go from raw Strand-seq reads, through mapping strand-state breakpoints. Before getting started, let's check if we have proper R environment first.

```
## As we finished plotting pipeline, let's go back to default R environment
vi ~/.Renviron

## Here, you can to back to default R environment by remove the path and save
ESC
:wq

## Let's check if we properly went back to default R environ

## Open R session
R

## Check library paths
.libPaths()

## Close R session
q()
n
```

2 Preparation of input files for breakpointR

Based on the quality checking of the overview plot, you may have selected 20 good libraries and 5 bad libraries. You can remove 5 bad libraries in the bam folder so that we can analyze breakpoint with good quality libraries.

```
## Let's go inside of the bam folder, and then remove the bad quality libraried
cd bam
```

3 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.

```
## Open Rstudio from the server



## 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
```

4 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.

```
## Let's go inside of the folder with output files from plotting pipeline
setwd('/scratch/name_abcd/Practical_WT/practical')


## Run breakpointR
breakpointr(inputfolder = "bam/",
  outputfolder = "BreakpointR_results",
  windowsize = 2000000,
  binMethod = 'size',
  pairedEndReads = TRUE,
  chromosomes = 'chr10',
  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
```

Next we will explore how to load USCS formatted bed files into the UCSC genome browser. You can download BreaksGraph to see the recurrent breakpoint and their locations in the UCSC genome browser. 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**.

```
## Download bedgraph files from the output folder
scp jeong@seneca:/path/to/breakpoints/BreakpointSummary_BreaksGraph.bed.gz /path/to/yourfolder/
```

Another important output is single-cell breakpoint plot. You can download single-cell breakpoint using the following command.

```
## Download pdf files from the output folder
scp jeong@seneca:/path/to/BreakpointR_results/plots/chr10_breakpoints.pdf /path/to/yourfolder/
```

To get the genomic position of breakpoints and confidence interval, we can explore following text file.

```
## Download pdf files from the output folder
less -S /path/to/BreakpointR_results/breakpoints/breakPointSummary.txt
```

5 Session Info

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

- R version 3.5.1 (2018-07-02), `x86_64-apple-darwin15.6.0`
- Locale: `C`
- Running under: `macOS High Sierra 10.13.3`
- Matrix products: default
- BLAS: `/System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/libBL`
- LAPACK: `/Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.26.0, GenomeInfoDb 1.16.0, GenomicRanges 1.32.7, IRanges 2.14.12, S4Vectors 0.18.3, breakpointR 1.0.0, breakpointRdata 1.0.0, cowplot 0.9.4, ggplot2 3.1.1, knitr 1.20
- Loaded via a namespace (and not attached): Biobase 2.40.0, BiocParallel 1.14.2, BiocStyle 2.8.2, Biostrings 2.50.2, DT 0.4, DelayedArray 0.6.6, DelayedMatrixStats 1.2.0, FNN 1.1.2.1, GenomeInfoDbData 1.1.0, GenomicAlignments 1.16.0, MOFAtools 0.99.0, Matrix 1.2-14, MultiAssayExperiment 1.6.0, R6 2.4.0, RColorBrewer 1.1-2, RCurl 1.95-4.12, Rcpp 1.0.1, Rhdf5lib 1.2.1, Rsamtools 1.34.0, SingleCellExperiment 1.2.0, SummarizedExperiment 1.10.1, XVector 0.20.0, assertthat 0.2.1, backports 1.1.4, beeswarm 0.2.3, bindr 0.1.1, bindrcpp 0.2.2, bitops 1.0-6, codetools 0.2-15, colorspace 1.4-1, compiler 3.5.1, corrplot 0.84, crayon 1.3.4, data.table 1.12.2, digest 0.6.19, doParallel 1.0.14, dplyr 0.7.6, dynamicTreeCut 1.63-1, edgeR 3.22.3, evaluate 0.11, foreach 1.4.4, ggbeeswarm 0.6.0, ggrepel 0.8.0, glue 1.3.1, grid 3.5.1, gridExtra 2.3, gtable 0.3.0, gtools 3.8.1, highr 0.7, htmltools 0.3.6, htmlwidgets 1.2, httpuv 1.5.1, igraph 1.2.2, iterators 1.0.10, jsonlite 1.6, later 0.8.0, lattice 0.20-35, lazyeval 0.2.2, limma 3.36.3, locfit 1.5-9.1, magrittr 1.5, matrixStats 0.54.0, mime 0.6, munsell 0.5.0, pheatmap 1.0.10, pillar 1.4.0, pkgconfig 2.0.2, plyr 1.8.4, promises 1.0.1, purrr 0.2.5, reshape2 1.4.3, reticulate 1.10, rhdf5 2.24.0, rjson 0.2.20, rlang 0.3.4, rmarkdown 1.10, rprojroot 1.3-2, rstudioapi 0.7, scales 1.0.0, scater 1.8.4, scran 1.8.4, shiny 1.3.2, shinydashboard 0.7.0, statmod 1.4.30, stringi 1.4.3, stringr 1.4.0, tibble 2.1.1, tidyr 0.8.1, tidyselect 0.2.4, tinytex 0.7, tools 3.5.1, tximport 1.8.0, vipor 0.4.5, viridis 0.5.1, viridisLite 0.3.0, withr 2.1.2, xfun 0.3, xtable 1.8-4, yaml 2.2.0, zlibbioc 1.26.0