A quick introduction to AneuFinder

Aaron Taudt*

June 1, 2016

Contents

*aaron.taudt@gmail.com

1 Introduction

AneuFinder offers functionality for the study of copy number variations (CNV) in whole-genome single cell sequencing (WGSCS) data. Functionality implemented in this package includes:

- CNV detection using a Hidden Markov Model on binned read counts.
- Various plotting capabilities like genomewide heatmaps of CNV state and arrayCGH-like plots.
- Export of CNV calls in BED format for upload to the UCSC genome browser.
- Quality metrics.
- Measures for addressing karyotype heterogeneity.

2 Quickstart

The main function of this package is called `Aneufinder`[1] and performs all the necessary steps to get from aligned reads to interpretable output:

```
Aneufinder(inputfolder='folder-with-BAM', outputfolder='output-directory',
           format='bam', numCPU=2)
```

Although in most cases the above command will produce reasonably good results, it might be worthwile to adjust the default parameters to improve performance and the quality of the results (see section **??**). You can get a description of all available parameters by typing

```
?Aneufinder
```

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

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

  ```
  Aneufinder(..., configfile='AneuFinder.config')
  ```

  to run another analysis with the same parameter settings.

[1] This function can also be run from command line, please see the INSTALL.md in the source package for details.

- binned: This folder contains the binned data. If you chose a correction method, you will also see a folder like 'binned-GC' in case of GC correction. You can load the data with

```
files <- list.files('output-directory/binned', full.names=TRUE)
binned.data <- loadGRangesFromFiles(files)
```

- browserfiles_data: This folder contains BED files with mapped reads that can be uploaded to the UCSC genome browser. The BED files contain the same reads as your input but filtered by mapping quality and other parameter settings that you can find in section [Binning] of the "AneuFinder.config" file.
- browserfiles: A folder which contains BED files with CNV calls that can be uploaded to the UCSC genome browser.
- data: This folder stores all the read data as RData objects. This exists mostly for internal usage.
- hmms: A folder with all produced Hidden Markov Models. You can load the results for further processing, such as quality control and customized plotting.

```
files <- list.files('output-directory/hmms', full.names=TRUE)
hmms <- loadHmmsFromFiles(files)
cl <- clusterByQuality(hmms)
heatmapGenomewide(cl$classification[[1]])
```

- plots: All plots that are produced by default will be stored here.

3 A detailed workflow

3.1 Mappability correction

The first step of your workflow should be the production of a reference file for mappability correction. Mappability correction is done via a variable-width binning approach (as compared to fixed-width bins) and requires a euploid reference. You can either simulate this reference file or take a real euploid reference. For optimal results we suggest to use a real reference, e.g. by merging BAM files of single cells from a euploid reference tissue. This can be achieved with the 'samtools merge' command (not part of R). Be careful: All CNVs that are present in the reference will lead to artifacts in the analysis later. This includes sex-chromosomes that are present in one copy only, so we advice to use a female reference and to exclude the Y-chromosome from the analysis. If you have no reference available, you can simulate one with the `simulateReads` command:

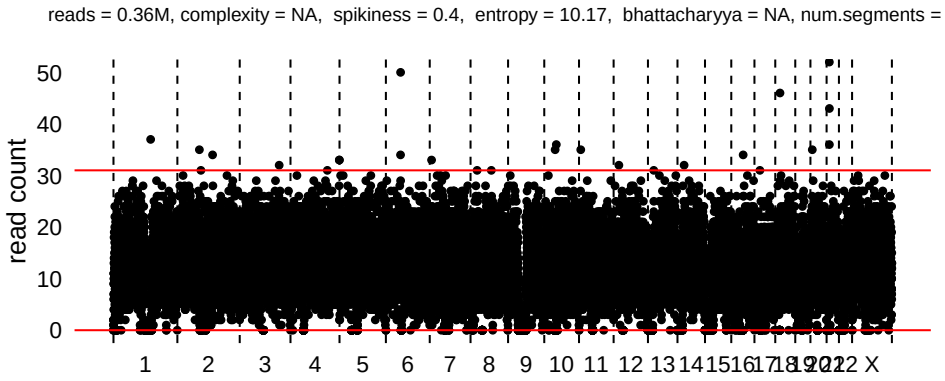
```
## Load human genome
library(BSgenome.Hsapiens.UCSC.hg19)

## Get a BAM file for the estimation of quality scores (adjust this to your experiment)
bamfile <- system.file("extdata", "BB150803_IV_074.bam", package="AneuFinderData")

## Simulate reads of length 51bp for human genome
# We simulate reads every 5000bp for demonstration purposes, for a real
# application you should use a much denser spacing (e.g. 500bp or less)
simulatedReads.file <- tempfile() # replace this with your destination file
simulateReads(BSgenome.Hsapiens.UCSC.hg19, readLength=51, bamfile=bamfile,
              file=simulatedReads.file, every.X.bp=5000)
```

This simulated FASTQ file must then be aligned with your aligner of choice (ideally the same that you used for your other samples) and given as reference in the `Aneufinder` function (option `variable.width.reference`).

3.2 Blacklisting

To further improve the quality of the results and remove artifacts caused by high mappability repeat regions, e.g. near centromers, a blacklist can be used in option `blacklist` of the `Aneufinder` function. All reads falling into the regions specified by the blacklist will be discarded when importing the read files. You can either download a blacklist from the UCSC genome browser, e.g. the "DAC Blacklisted Regions from ENCODE/DAC(Kundaje)" mappability track, or make your own. For optimal results, we advice to make your own blacklist from a euploid reference. The following code chunck takes a euploid reference and makes fixed-width bins of 100kb. Bins with read count above and beloow the 0.999 and 0.05 quantile are taken as blacklist:

```
## Get a euploid reference (adjust this to your experiment)
bedfile <- system.file("extdata", "hg19_diploid.bam.bed.gz", package="AneuFinderData")

## Make 100kb fixed-width bins
bins <- binReads(bedfile, format='bed', assembly='hg19',
                 binsizes=100e3, chromosomes=c(1:22,'X'))[[1]]
## Make a plot for visual inspection and get the blacklist
lcutoff <- quantile(bins$counts, 0.05)
ucutoff <- quantile(bins$counts, 0.999)
p <- plot(bins) + coord_cartesian(ylim=c(0,50))
p <- p + geom_hline(aes(yintercept=lcutoff), color='red')
p <- p + geom_hline(aes(yintercept=ucutoff), color='red')
print(p)
```




```
blacklist <- bins[bins$counts < ucutoff & bins$counts > lcutoff]
## Write blacklist to file
blacklist.file <- tempfile()
exportGRanges(blacklist, filename=blacklist.file, header=FALSE,
              chromosome.format='NCBI')
```

3.3 Running Aneufinder

The function **Aneufinder** takes an input folder with BAM or BED files and produces an output folder with results, plots and browserfiles. The following code is an example of how to run **Aneufinder** with variable-width bins (see section **??**), blacklist (see section **??**) and GC-correction. Results will be stored in 'outputfolder/hmms' as RData objects for further processing such as quality filtering and customized plotting.

```
## First, get some data and reference files (adjust this to your experiment)
var.width.ref <- system.file("extdata", "hg19_diploid.bam.bed.gz", package="AneuFinderData")
blacklist <- system.file("extdata", "blacklist-hg19.bed.gz", package="AneuFinderData")
datafolder <- system.file("extdata", "B-ALL-B", package = "AneuFinderData")
list.files(datafolder) # only 3 cells for demonstration purposes

## [1] "MB140210_I_003.bam.bed.gz" "MB140210_I_004.bam.bed.gz"
## [3] "MB140210_I_005.bam.bed.gz"

## Library for GC correction
library(BSgenome.Hsapiens.UCSC.hg19)

## Produce output files
cnv.states <- c('zero-inflation', paste0(1:10, '-somy'))
outputfolder <- tempdir()
Aneufinder(inputfolder = datafolder, outputfolder = outputfolder, format = 'bed',
           numCPU = 3, binsizes = c(5e5, 1e6), variable.width.reference = var.width.ref,
           chromosomes = c(1:22,'X','Y'), blacklist = blacklist, states = cnv.states,
           correction.method = 'GC', GC.BSgenome = BSgenome.Hsapiens.UCSC.hg19,
           num.trials = 1, assembly = 'hg19')
```

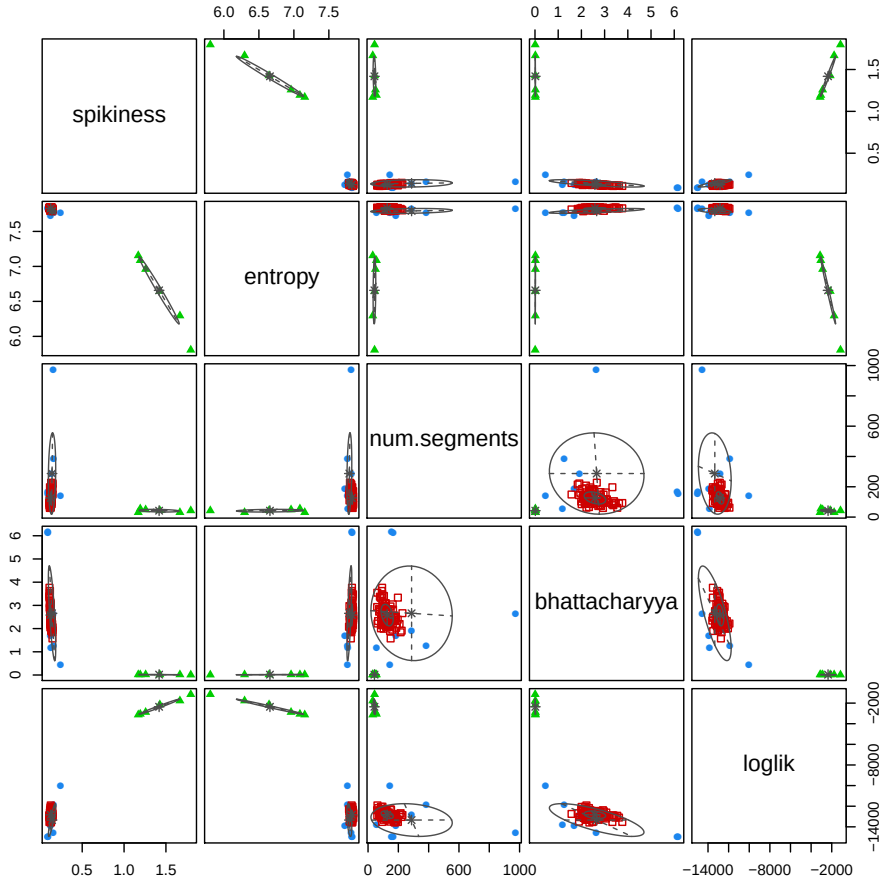
```
## Setting up parallel execution with 3 CPUs ...
##  0.69s
## Fetching chromosome lengths from UCSC ...
## Warning in FUN(genome = names(SUPPORTED_UCSC_GENOMES)[idx], circ_seqs = supported_genome$circ_seqs,
:  NCBI seqlevel was set to NA for hg19 UCSC seqlevel(s) not in the
##  NCBI assembly:  chrM
##  0.86s
## ==> Making bins:
## Reading file hg19_diploid.bam.bed.gz ...
##  4.86s
## Filtering reads ...
##  0.73s
## Filtering blacklisted regions ...
##  0.54s
## Binning reads in fixed-width windows ...
##  3.98s
## Making variable-width windows for bin size 5e+05 ...
##  1.45s
## Making variable-width windows for bin size 1e+06 ...
##  1.49s
## ==| Finished making bins.
## Binning the data ...
##  21.2s
## Saving reads as .RData ...
##  0.01s
## Exporting data as browser files ...
##  5.54s
## GC correction ...
##  56.47s
## Running univariate HMMs ...
##  20.92s
## Plotting genomewide heatmaps ...
##  1.67s
## Plotting chromosome heatmaps ...
##  2.93s
## Making profile and distribution plots ...
##  3.17s
## Exporting browser files ...
##  0.76s
## ==> Total time spent:  129s <==
```

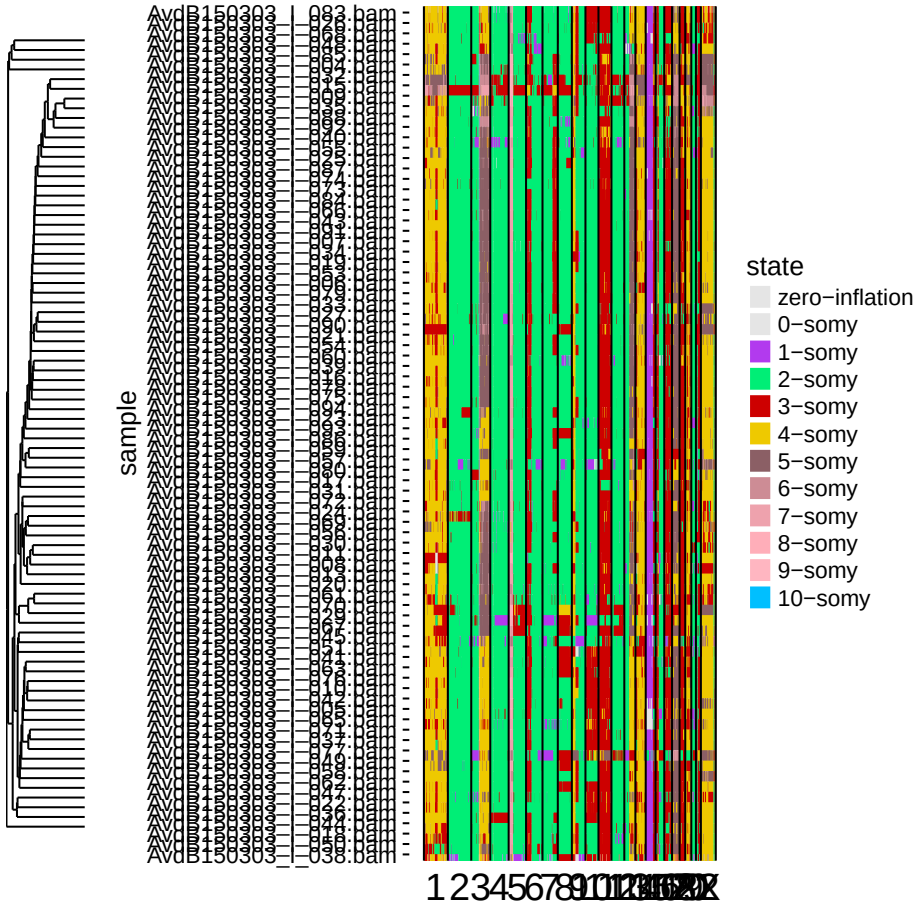
3.4 Quality control

Once the function `Aneufinder` has completed, results will be accessible as .RData files under 'outputfolder/hmms'. Single cell sequencing is prone to noise and therefore it is a good idea to filter the results by quality to retain only high-quality cells. We found that simple filtering procedures such as cutoffs on the total number of reads etc., are insufficient to distinguish good from bad-quality libraries. Therefore, we have implemented a multivariate clustering approach that works on multiple quality metrics (see ?clusterByQuality for details) to increase robustness of the filtering. Here is an

example demonstrating the usage of `clusterByQuality`.

```
## Get some pre-produced results (adjust this to your experiment)
results <- system.file("extdata", "primary-lung", "hmms", package="AneuFinderData")
files <- list.files(results, full.names=TRUE)

## Cluster by quality
cl <- clusterByQuality(files)
plot(cl$Mclust, what='classification')
print(cl$parameters)

##      spikiness  entropy num.segments bhattacharyya     loglik
## [1,] 0.1246761 7.827808    126.25308    2.58241089 -12834.221
## [2,] 0.1396946 7.792930    287.37629    2.65826609 -13342.096
## [3,] 1.4186789 6.657955     42.16667    0.01576416  -2356.649

## Apparently, the third cluster corresponds to failed libraries
## while the first cluster contains high-quality libraries

## Select the best cluster and plot it
selected.files <- unlist(cl$classification[[1]])
heatmapGenomewide(selected.files)
```


spikiness
entropy
num.segments
bhattacharyya
loglik
6.0 6.5 7.0 7.5
0 1 2 3 4 5 6
0.5 1.0 1.5
0 200 600 1000
−14000 −8000 −2000



3.5 Karyotype measures

This package implements two measures to quantify karyotype heterogeneity, an *aneuploidy* and a *heterogeneity* score. Both measures are independent of the number of cells, the length of the genome, and take into account every position in the genome. The following example compares the heterogeneity and aneuploidy between a primary lung cancer and the corresponding liver metastasis.

```
## Get some pre-produced results (adjust this to your experiment)
results <- system.file("extdata", "primary-lung", "hmms", package="AneuFinderData")
files.lung <- list.files(results, full.names=TRUE)
results <- system.file("extdata", "metastasis-liver", "hmms", package="AneuFinderData")
files.liver <- list.files(results, full.names=TRUE)
```

```
## Get karyotype measures
k.lung <- karyotypeMeasures(files.lung)
k.liver <- karyotypeMeasures(files.liver)

## Print the scores in one data.frame
df <- rbind(lung = k.lung$genomewide, liver = k.liver$genomewide)
print(df)

##       Aneuploidy Heterogeneity
## lung   0.9875727     0.4732248
## liver  0.9107400     0.2633423

## While the aneuploidy is similar between both cancers, the heterogeneity is
## nearly twice as high for the primary lung cancer.
```

4 Session Info

```
sessionInfo()

## R version 3.3.0 (2016-05-03)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.9.5 (Mavericks)
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] BSgenome.Hsapiens.UCSC.hg19_1.4.0 BSgenome_1.40.0
##  [3] rtracklayer_1.32.0                Biostrings_2.40.1
##  [5] XVector_0.12.0                    AneuFinder_1.0.3
##  [7] AneuFinderData_1.0.2              cowplot_0.6.2
##  [9] ggplot2_2.1.0                     GenomicRanges_1.24.0
## [11] GenomeInfoDb_1.8.2                IRanges_2.6.0
## [13] S4Vectors_0.10.1                  BiocGenerics_0.18.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.5                compiler_3.3.0
##  [3] highr_0.6                  formatR_1.4
##  [5] plyr_1.8.3                 bitops_1.0-6
##  [7] iterators_1.0.8            tools_3.3.0
##  [9] zlibbioc_1.18.0            digest_0.6.9
## [11] mclust_5.2                 evaluate_0.9
## [13] gtable_0.2.0               foreach_1.4.3
## [15] polynom_1.3-8              ggdendro_0.1-20
## [17] preseqR_2.0.0              stringr_1.0.0
## [19] knitr_1.13                 caTools_1.17.1
## [21] gtools_3.5.0               grid_3.3.0
```

```
## [23] Biobase_2.32.0             XML_3.98-1.4
## [25] BiocParallel_1.6.2         gdata_2.17.0
## [27] reshape2_1.4.1             magrittr_1.5
## [29] ReorderCluster_1.0         scales_0.4.0
## [31] Rsamtools_1.24.0           gplots_3.0.1
## [33] codetools_0.2-14           MASS_7.3-45
## [35] GenomicAlignments_1.8.0    SummarizedExperiment_1.2.2
## [37] colorspace_1.2-6           labeling_0.3
## [39] KernSmooth_2.23-15         stringi_1.1.1
## [41] RCurl_1.95-4.8             doParallel_1.0.10
## [43] munsell_0.4.3

warnings()

## NULL
```