jDMR: a heuristic DMR caller for population-level WGBS data

Rashmi Hazarika, Y.Shahryary & Frank Johannes

2021-06-18

Contents

1 Identification of cytosine clusters

jDMR detects DMRs using two approaches a) finding cytosine clusters in the Genome b) using a binning approach. You can use either/or both methods to obtain the region calls. The remaining steps, makeDMRmatrix, filterDMRmatrix are the same for both methods.

```
library(jDMR)
```

```
out.dir <- "/myfolder/DMR-results"
```

1.1 Extract cytosines from FASTA and generate cytosine clusters

Skip this step if you want to run the grid approach for DMR calling. Go to section 2.1, 2.3, 2.4.

Read in the reference genome FASTA file. Here, we will work with chromosome 1 from *Arabidopsis thaliana*.

```
fasta <- system.file("extdata", "Arabidopsis_thaliana.TAIR10.dna.chromosome.1.fa.gz",
    package = "jDMR")
myfasta <- readDNAStringSet(fasta)
```

Run "CfromFASTAv4" function for chromosome 1. This function extracts cytosines from FASTA and generates the output file "cytosine_positions_chr1.csv".

```
CfromFASTAv4(fasta = myfasta, chr = 1, out.dir = out.dir, write.output = TRUE)
```

```
  - Processing chr: 1
  -------------------------
  - Converting DNA .....
  - Scanning CG + strand .....
  - Scanning CHG + strand .....
  - Scanning CHH + strand .....
  - Scanning CG - strand .....
  - Scanning CHG - strand .....
  - Scanning CHH - strand .....
  - Combining contexts .....
  - Writing out file .....
```

Run "makeReg". This function will call "cytosine_positions_chr1.csv" and extract cytosines clusters for "CG" context

```
ref.genome <- fread(paste0(out.dir, "/cytosine_positions_chr", 1, ".csv", sep = ""))

makeReg(ref.genome = ref.genome, contexts = c("CG", "CHG", "CHH"), makeRegnull = c(FALSE),
    chr = 1, min.C = 5, N.boot = 10^5, N.sim.C = "all", fp.rate = 0.01, set.tol = 0.01,
    out.dir = out.dir, out.name = "Arabidopsis")
```

If you want to run for all chromosomes together, combine the two functions "CfromFASTAv4" and "makeReg" into one single script and execute it:

Refer to script, RUN_makeReg.R for the code.

```
out.name <- "Arabidopsis"
contexts <- c("CG", "CHG", "CHH")
makeNull <- c(TRUE, TRUE, TRUE)
min.C <- 5
fp.rate <- 0.01

wd <- "/myfolder/fasta-files"
```

```
# Supply all FASTA files in one folder
chrfiles <- list.files(paste0(wd, "FASTA"), pattern = paste0("*.fa.gz$"), full.names = TRUE)

# I am creating a new folder 'min.C_5' here
if (!dir.exists(paste0(out.dir, "min.C_5"))) {
    cat(paste0("Creating directory "))
    dir.create(paste0(out.dir, "min.C_5"))
} else {
    cat("directory exists!")
}

for (i in 1:length(chrfiles)) {
    fasta <- readDNAStringSet(chrfiles[i])
    chr <- gsub(".*chromosome.|\\.fa.gz$", "", basename(chrfiles[i]))
    cat(paste0("Running for chr:", chr, "\n"), sep = "")

    # extract cytosines from Fasta
    system.time(CfromFASTAv4(fasta = fasta, chr = chr, out.dir = paste0(out.dir,
        "min.C_5/"), write.output = TRUE))

    # Calling regions; calls the file created by CfromFASTAv4
    ref.genome <- fread(paste0(out.dir, "min.C_5/cytosine_positions_chr", chr, ".csv",
        sep = ""))

    system.time(makeReg(ref.genome = ref.genome, contexts = contexts, makeRegnull = makeNull,
        chr = chr, min.C = min.C, N.boot = 10^5, N.sim.C = "all", fp.rate = fp.rate,
        set.tol = 0.01, out.dir = paste0(out.dir, "min.C_5/"), out.name = out.name))
}
```

1.2 Output files

Output file "Arabidopsis_regions_chr1_CG.Rdata" is a Rdata file which has the following structure.

```
head(regionfile$reg.obs)
```

```
    chr start   end cluster.length region
1   1  3696  3856            160   reg1
2   1 12100 12155             55   reg2
3   1 20991 21026             35   reg3
4   1 21257 21293             36   reg4
5   1 29966 30008             42   reg5
6   1 46099 46141             42   reg6
```

2 Generation of Cytosine region-level calls

2.1 Input files

For generation of region-level calls, jDMR requires the following inputs.

2.1.1 Methimpute files:

Full PATH of base-level methylome outputs (generated using the R package "Methimpute") should be specified in the file "listFiles1.fn". A column called "sample" should contain any assigned name.

```
samplefile1 <- system.file("extdata", "listFiles1.fn", package = "jDMR")
fread(samplefile1, header = TRUE)
```

```
                                file     sample
  1: methimpute-out/methylome_A_All.txt methylomeA
  2: methimpute-out/methylome_B_All.txt methylomeB
  3: methimpute-out/methylome_C_All.txt methylomeC
  4: methimpute-out/methylome_D_All.txt methylomeD
  5: methimpute-out/methylome_E_All.txt methylomeE
  6: methimpute-out/methylome_F_All.txt methylomeF
```

file: full PATH of file

sample: a sample name

For pairwise control-treatment data-sets with replicates, an additional column "replicate" should be provided. See structure below.

```
samplefile2 <- system.file("extdata", "listFiles2.fn", package = "jDMR")
fread(samplefile2, header = TRUE)
```

```
                                file  sample replicate
  1: methimpute-out/methylome_A_All.txt    Col0      rep1
  2: methimpute-out/methylome_B_All.txt    Col0      rep2
  3: methimpute-out/methylome_C_All.txt mutant1      rep1
  4: methimpute-out/methylome_D_All.txt mutant1      rep2
  5: methimpute-out/methylome_E_All.txt mutant2      rep1
  6: methimpute-out/methylome_F_All.txt mutant2      rep2
```

file: full PATH of file

sample: a sample name

replicate: label for replicates

2.1.2 Cytosine region files (Optional, only if you will run "runMethimputeRegions") :

These files containing cytosine clusters were generated using the function "makeReg". See section 1.1

```
Regionsfolder <- system.file("extdata", "min.C_5/fp0.01", package = "jDMR")
```

2.2 Run Methimpute for cytosine regions

Run function "runMethimputeRegions" on identified cytosine clusters.

```
runMethimputeRegions(Regionfiles = Regionsfolder, samplefiles = samplefile1, genome = "Arabidopsis",
    context = c("CG", "CHG", "CHH"), out.dir = out.dir)
```

2.3 Run Methimpute on a binned genome.

For a non-sliding window approach use window size=100 and step size=100. Useful for a) mSFS(maybe) b) region-level epimutation estimations

For a sliding-window approach use window size=100 and step size=50. Useful for a) meQTL mapping b) DMR calling across treatments c) DMRs in populations

```
fasta.files <- system.file("extdata", package = "jDMR")

runMethimputeGrid(fasta = fasta.files, samplefiles = samplefile1, genome = "Arabidopsis",
    context = c("CG"), out.dir = out.dir, win = 100, step = 100, mincov = 0, nCytosines = 5)
```

2.4 Output files

"region-level methylome files" have the following structure

```
head(region.file)
```

```
    seqnames start  end context posteriorMax status rc.meth.lvl
 1:        1   101  200      CG            1      M     0.75833
 2:        1   601  700      CG            1      M     0.75833
 3:        1   901 1000      CG            1      M     0.75833
 4:        1  2401 2500      CG            1      U     0.00711
 5:        1  2801 2900      CG            1      U     0.00711
 6:        1  2901 3000      CG            1      U     0.00711
```

seqnames, start and strand: Chromosome coordinates

context: Sequence context of cytosine i.e CG,CHG,CHH

posteriorMax: Posterior value of the methylation state call

status : Methylation status

rc.meth.lvl: Recalibrated methylation level calculated from the posteriors and fitted parameters

3 Generate DMR matrix

3.1 Run "makeDMRmatrix"

"makeDMRmatrix" function generates 1) binary matrix (0,1) and 2) matrix of rc.meth.lvls for all samples in one dataframe.

```
makeDMRmatrix(context = c("CG", "CHG", "CHH"), samplefiles = samplefile1, input.dir = out.dir,
    out.dir = out.dir)
```

3.2 Output files

"CG_StateCalls.txt" has the following structure. "0" in the output matrix denotes "Unmethylated" and "1" stands for "Methylated".

```
statecalls <- fread(paste0(out.dir, "CG_StateCalls.txt", sep = ""), header = TRUE)
head(statecalls)
```

```
    seqnames start  end GSM2328622 GSM2328623 GSM2328624
 1:        1   101  200          1          1          1
 2:        1   601  700          1          1          1
 3:        1   901 1000          1          1          1
 4:        1  2401 2500          0          0          0
 5:        1  2801 2900          0          0          0
 6:        1  2901 3000          0          0          0
```

"CG_rcMethlvl.txt" has the following structure. The output matrix contains recalibrated methylation levels for each sample and for the specific region.

```
rcmethlvls <- fread(paste0(out.dir, "CG_rcMethlvl.txt", sep = ""), header = TRUE)
head(rcmethlvls)
```

```
    seqnames start  end GSM2328622 GSM2328623 GSM2328624
 1:        1   101  200    0.75833    0.78099    0.78311
 2:        1   601  700    0.75833    0.78099    0.78311
 3:        1   901 1000    0.75833    0.78099    0.78311
 4:        1  2401 2500    0.00711    0.00945    0.00928
 5:        1  2801 2900    0.00711    0.00945    0.00928
 6:        1  2901 3000    0.00711    0.00945    0.00928
```

4 Filter DMR matrix

4.1 Filter the DMR matrix with the following options

"filterDMRmatrix" function filters "CG_StateCalls.txt" and "CG_rcMethlvl.txt" for non-polymorphic patterns by default.

epiMAF.cutoff parameter can be used for population level data. This option can be used to filter for Minor Epi-Allele frequency as specified by user (e.g 0.33). By default, this option is set to NULL.

replicate.consensus option can be used for pairwise control-treatment data-sets with replicates. With the *replicate.consensus*, user can specify the percentage of concordance in methylation states in samples with multiple replicates. For datasets with just 2 replicates, *replicate.consensus* should be set as 1 (means 100% concordance). By default, this option is set to NULL.

grid.DMR if you used the grid approach to call DMRs set to TRUE otherwise set to FALSE. The output will contain merged regions.

```
## Please run filterDMRmatrix function based on the type of data you have.

filterDMRmatrix(gridDMR = TRUE, data.dir = out.dir)
# replicate.consensus=8 epiMAF.cutoff=0.33
```

4.2 Filtered Output

"CG_StateCalls-filtered.txt" has the following structure.

```
statecallsFiltered <- fread(paste0(out.dir, "CG_StateCalls-filtered.txt", sep = ""),
    header = TRUE)
head(statecallsFiltered)
```

```
    seqnames start   end GSM2328622 GSM2328623 GSM2328624
 1:        1  5101  5200          0          0          1
 2:        1  5401  5500          0          1          1
 3:        1 10101 10200          0          1          0
 4:        1 18701 18800          1          0          1
 5:        1 25101 25200          1          0          1
 6:        1 25301 25400          1          0          1
```

5 Annotate DMRs

Multiple gff3 annotation files can be supplied as a vector with the *gff* option. Single/multiple files containing filtered DMR matrix should be provided with the *file.list* option. If you are following the grid approach then supply "CG_StateCalls-filtered-merged.txt"

```
# annotation files
gff.AT <- "/Annotations/Arabidopsis_thaliana.TAIR10.47.gff3"
gff.TE <- "/Annotations/TAIR10_TE.gff3"
gff.pr <- "/Annotations/TAIR10_promoters.gff3"

# Please supply the text files to be annotated in a separate folder.  For e.g I
# make a new folder 'mysamples'. In the case of gridDMR supply the (*merged.txt)
# files by moving them to 'mysamples' folder
mydir <- paste0(out.dir, "mysamples")

# you can specify the following available annotations. if you have your custom
# file let me know.
#'chromosome','gene','mRNA','five_prime_UTR','exon','CDS',
#'three_prime_UTR','ncRNA_gene','lnc_RNA','miRNA','tRNA','ncRNA',
#'snoRNA','snRNA','rRNA','TE','promoters'

annotateDMRs(gff.files = c(gff.AT, gff.TE, gff.pr), annotation = c("gene", "promoters",
    "TE"), input.dir = out.dir, gff3.out = TRUE, out.dir = out.dir)
```

5.1 Output files

Mapped files are output in gff3 format. Addiitonally, a DMR count table is generated.

```
DMRcount <- fread(paste0(out.dir, "mysamples/DMR-counts.txt", sep = ""), header = TRUE)

DMRcount
```

```
                    sample total.DMRs gene promoters    TE multiple.overlaps
 1:  CG_StateCalls-filtered       5428 1900       390  1407              1129
 2: CHG_StateCalls-filtered       2528  359       168  1300               365
 3: CHH_StateCalls-filtered      52782 5731      3845 28001              5625
```

6 R session info

```
sessionInfo()
```

```
R version 4.0.1 (2020-06-06)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS  10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] parallel  stats4    stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] data.table_1.14.0    Biostrings_2.58.0    XVector_0.30.0       jDMR_0.1.0           GenomicRanges_1.42.0
[6] GenomeInfoDb_1.26.4  IRanges_2.24.1       S4Vectors_0.28.1     BiocGenerics_0.36.0

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.6                 lattice_0.20-41            prettyunits_1.1.1          Rsamtools_2.6.0
```

```
 [5] assertthat_0.2.1           digest_0.6.27              utf8_1.2.1                 R6_2.5.0
 [9] plyr_1.8.6                 evaluate_0.14              ggplot2_3.3.3              pillar_1.5.1
[13] progress_1.2.2             zlibbioc_1.36.0            rlang_0.4.10               R.utils_2.10.1
[17] R.oo_1.24.0                Matrix_1.3-2               rmarkdown_2.7              BiocParallel_1.24.1
[21] stringr_1.4.0              RCurl_1.98-1.3             munsell_0.5.0              DelayedArray_0.16.3
[25] compiler_4.0.1             rtracklayer_1.50.0         xfun_0.22                  pkgconfig_2.0.3
[29] htmltools_0.5.1.1          tidyselect_1.1.0           SummarizedExperiment_1.20.0 tibble_3.1.0
[33] GenomeInfoDbData_1.2.4     matrixStats_0.58.0         XML_3.99-0.6               fansi_0.4.2
[37] crayon_1.4.1               dplyr_1.0.5                MASS_7.3-53.1              GenomicAlignments_1
[41] bitops_1.0-6               R.methodsS3_1.8.1          grid_4.0.1                 gtable_0.3.0
[45] lifecycle_1.0.0            DBI_1.1.1                  magrittr_2.0.1             formatR_1.8
[49] scales_1.1.1               stringi_1.5.3              reshape2_1.4.4             seqinr_4.2-5
[53] ellipsis_0.3.1             generics_0.1.0             vctrs_0.3.6                methimpute_1.12.0
[57] tools_4.0.1                ade4_1.7-16                Biobase_2.50.0             glue_1.4.2
[61] purrr_0.3.4                hms_1.0.0                  MatrixGenerics_1.2.1       yaml_2.2.1
[65] colorspace_2.0-0           minpack.lm_1.2-1           knitr_1.31
```