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

Rashmi Hazarika, Y.Shahryary & Frank Johannes

2021-04-15

Contents

1 Identification of cytosine clusters

The first steps for running jDMR are to load all required libraries, set PATHs of working, output directory. jDMR detects DMRs using two approaches a) finding cytosine clusters in the Genome b) using a binning approach. You can use either/or methods to obtain the region calls. The remaining steps, makeDMRmatrix, filterDMRmatrix are the same for both methods.

```
library(data.table)
library(dplyr)
library(stringi)
library(stringr)
library(Biostrings)
library(methimpute)
library(rtracklayer)
library(tidyr)
```

```
wd <- "/myfolder/DMR-Analysis"
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, 2.2, 2.4.

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

```
# Load source code
source(paste0(wd, "/CfromFASTAv4.R", sep = ""))
fasta <- readDNAStringSet(paste0(wd, "FASTA/Arabidopsis_thaliana.TAIR10.dna.chromosome.1.fa.gz",
    sep = ""))
```

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

```
CfromFASTAv4(fasta = fasta, 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

```
#Load source code
source(paste0(wd, "/makeReg.R", sep=""))

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

makeReg(ref.genome=ref.genome,
        contexts=c("CG"), # all contexts can be specified as ("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.

```
source(paste0(wd, "/CfromFASTAv4.R", sep = ""))
source(paste0(wd, "/makeReg.R", sep = ""))

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

# 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 <- paste0(wd, "/listFiles1.fn", sep = "")
fread(samplefile1, header = TRUE)
```

```
                                        file    sample
  1: /jlab/data/methimpute-out/methylome_A_All.txt mysampleA
  2: /jlab/data/methimpute-out/methylome_B_All.txt mysampleB
  3: /jlab/data/methimpute-out/methylome_C_All.txt mysampleC
  4: /jlab/data/methimpute-out/methylome_D_All.txt mysampleD
  5: /jlab/data/methimpute-out/methylome_E_All.txt mysampleE
  6: /jlab/data/methimpute-out/methylome_F_All.txt mysampleF
```

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 <- paste0(wd, "/listFiles-replicates.fn", sep = "")
fread(samplefile2, header = TRUE)
```

```
                                     file  sample replicate
  1: /jlab/data/methimpute-out/methylome_A.txt      WT      rep1
  2: /jlab/data/methimpute-out/methylome_B.txt      WT      rep2
  3: /jlab/data/methimpute-out/methylome_C.txt mutant1      rep1
  4: /jlab/data/methimpute-out/methylome_D.txt mutant1      rep2
  5: /jlab/data/methimpute-out/methylome_E.txt mutant2      rep1
  6: /jlab/data/methimpute-out/methylome_F.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 <- paste0(wd, "min.C_5/")
```

2.2 Load the source code

```
# Load source code
source(paste0(wd, "/globFun.R", sep = ""))
source(paste0(wd, "/MethimputeReg.R", sep = ""))
source(paste0(wd, "/runMethimpute.R", sep = ""))
```

2.3 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 = myoutput)
```

2.4 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

```
# If your genome is Arabidopsis you can use the ones provided.
# If your genome is not Arabidopsis, supply your custom Chr lengths.
# (refer to RUN_jDMR.R script on how to obtain chrlengths from fasta index file)

chrs <- c(chr1=30427671, chr2=19698289, chr3=23459830, chr4=18585056, chr5=26975502)

runMethimputeGrid(out.dir=myoutput,
                  scaffold=FALSE, #if your genome has scaffolds set to TRUE
                  chrfile=chrlengths,
                  win=100,
                  step=100,
                  genome="Arabidopsis",
                  samplefiles=samplefile1,
                  mincov=0,
                  nCytosines=5,
                  context=c("CG","CHG","CHH"))
```

2.5 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.

```
# load source code
source(paste0(wd, "/makeDMRmatrix.R", sep = ""))

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 mysampleA mysampleB mysampleC
 1:        1  3696  3856         0         0         0
 2:        1 12100 12155         0         0         0
 3:        1 20991 21026         0         0         0
 4:        1 21257 21293         0         0         0
 5:        1 29966 30008         1         1         1
 6:        1 46099 46141         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 mysampleA mysampleB mysampleC
 1:        1  3696  3856   0.00580   0.00633   0.00608
 2:        1 12100 12155   0.00580   0.00633   0.00608
 3:        1 20991 21026   0.00580   0.00633   0.00608
 4:        1 21257 21293   0.00580   0.00633   0.00608
 5:        1 29966 30008   0.82113   0.83046   0.82797
 6:        1 46099 46141   0.00580   0.00633   0.00608
```

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). Otherwise, this option should be 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). Otherwise, this option should be 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.

```
# load source code
source(paste0(wd, "/globFun.R", sep = ""))
source(paste0(wd, "/filterDMRmatrix.R", sep = ""))

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

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

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 mysampleA mysampleB mysampleC
  1:        1  95240  95276         0         1         1
  2:        1 212502 212535         0         0         1
  3:        1 213577 213616         1         0         1
  4:        1 216237 216268         1         0         0
  5:        1 359705 359740         1         0         0
  6:        1 360106 360143         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"

```
# Load source code
source(paste0(wd, "/annotateDMRs.R", sep = ""))

# 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 = mydir, gff3.out = TRUE, out.dir = mydir)
```

5.1 Output files

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

```
annotedOut <- import.gff3(paste0(out.dir, "mysamples/CG_rcMethlvl-filtered-merged_annotation.gff3",
    sep = ""), colnames = c("source", " type", "annotation", "ID", "region"))

annotedOut
```

```
  GRanges object with 6401 ranges and 5 metadata columns:
           seqnames            ranges strand |      source        type  annotation             ID      region
              <Rle>         <IRanges>  <Rle> |    <factor> <character> <character>    <character> <character>
       [1]        1       95240-95276      * | rtracklayer        <NA>        gene gene:AT1G01220         DMR
       [2]        1     212502-212535      * | rtracklayer        <NA>        gene gene:AT1G01580         DMR
       [3]        1     216237-216268      * | rtracklayer        <NA>        gene gene:AT1G01590         DMR
       [4]        1     359705-359740      * | rtracklayer        <NA>        gene gene:AT1G02050         DMR
       [5]        1     360106-360143      * | rtracklayer        <NA>        gene gene:AT1G02050         DMR
       ...      ...               ...    ... .         ...         ...         ...            ...         ...
    [6397]        5 24601855-24601893      * | rtracklayer        <NA>          TE     AT5TE88530         DMR
    [6398]        5 26015707-26015741      * | rtracklayer        <NA>          TE     AT5TE93635         DMR
    [6399]        5 26114751-26114806      * | rtracklayer        <NA>          TE     AT5TE94030         DMR
    [6400]        5 26219371-26219687      * | rtracklayer        <NA>          TE     AT5TE94410         DMR
    [6401]        5 26606871-26606908      * | rtracklayer        <NA>          TE     AT5TE95830         DMR
    -------
    seqinfo: 5 sequences from an unspecified genome; no seqlengths
```

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

DMRcount
```

```
                          sample total.DMRs gene promoters   TE multiple.overlaps
  1: CG_rcMethlvl-filtered-merged       5428 1900       390 1407              1129
```

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] stats4    parallel  stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] tidyr_1.1.3          rtracklayer_1.50.0   methimpute_1.12.0    ggplot2_3.3.3        GenomicRanges_1.42.
 [6] GenomeInfoDb_1.26.4  Biostrings_2.58.0    XVector_0.30.0       IRanges_2.24.1       S4Vectors_0.28.1
[11] BiocGenerics_0.36.0  stringr_1.4.0        stringi_1.5.3        dplyr_1.0.5          data.table_1.14.0

loaded via a namespace (and not attached):
 [1] SummarizedExperiment_1.20.0 minpack.lm_1.2-1           tidyselect_1.1.0           xfun_0.22
 [5] purrr_0.3.4                 reshape2_1.4.4             lattice_0.20-41            colorspace_2.0-0
 [9] vctrs_0.3.6                 generics_0.1.0             htmltools_0.5.1.1          yaml_2.2.1
[13] utf8_1.2.1                  XML_3.99-0.6               rlang_0.4.10               pillar_1.5.1
[17] glue_1.4.2                  withr_2.4.1                DBI_1.1.1                  BiocParallel_1.24.1
[21] matrixStats_0.58.0          GenomeInfoDbData_1.2.4     lifecycle_1.0.0            plyr_1.8.6
[25] MatrixGenerics_1.2.1        zlibbioc_1.36.0            munsell_0.5.0              gtable_0.3.0
[29] evaluate_0.14               Biobase_2.50.0             knitr_1.31                 fansi_0.4.2
[33] Rcpp_1.0.6                  scales_1.1.1               formatR_1.8                DelayedArray_0.16.3
[37] Rsamtools_2.6.0             digest_0.6.27              grid_4.0.1                 tools_4.0.1
[41] bitops_1.0-6                magrittr_2.0.1             RCurl_1.98-1.3             tibble_3.1.0
[45] crayon_1.4.1                pkgconfig_2.0.3            Matrix_1.3-2               ellipsis_0.3.1
[49] assertthat_0.2.1            rmarkdown_2.7              R6_2.5.0                   GenomicAlignments_1
[53] compiler_4.0.1
```