

jDMRgrid: a heuristic DMR caller for WGBS data using grid approach

Robert Piecyk, Rashmi Hazarika, Yadi Shahryary, Frank Johannes

2023-07-14

Contents

1	Input files	2
1.1	Methimpute files:	2
1.2	A metadata file containing description about samples	2
2	Step 1: Generate cytosine region calls from genome	3
2.1	Run jDMRgrid on a binned genome	3
2.1.1	Output files of jDMR Grid approach	3
3	Step 2: Generate DMR matrix	5
3.1	Run “makeDMRmatrix”	5
3.2	Output files of DMRmatrix function	5
3.3	Split DMR matrix into pairwise groups (only applicable for datasets with control- treatments)	6
4	Step 3: Filter DMR matrix	7
4.1	Filter the DMR matrix	7
4.2	Filtered Output	7
5	Step 4: Search for context-specific and annotate DMRs	8
5.1	Output context specific DMRs	8
5.2	Annotate DMRs	8
5.3	Output files after annotation	8
6	R session info	10

1 Input files

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

1.1 Methimpute files:

Base-level methylome outputs (generated using the R package “Methimpute”)

1.2 A metadata file containing description about samples

For population data-sets without replicates, listfiles.fn should have the structure below.

file: full PATH of file.

sample: a sample 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

For pairwise control-treatment data-sets with replicates, additional columns “replicate” and “group” should be provided. See structure below.

file: full PATH of file

sample: a sample name

replicate: label for replicates

group: label for control and treatment groups

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

	file	sample	replicate	group
1:	toyData/methimpute_p1.csv	WT	rep1	control
2:	toyData/methimpute_p2.csv	WT	rep2	control
3:	toyData/methimpute_p3.csv	mutant1	rep1	treatment1
4:	toyData/methimpute_p4.csv	mutant1	rep2	treatment1

2 Step 1: Generate cytosine region calls from genome

jDMR detects DMRs using two approaches a) finding cytosine clusters in the genome (section 2.1) b) using a binning approach (section 2.2). You can use either of the methods to obtain the region calls. The remaining steps, makeDMRmatrix, filterDMRmatrix, annotateDMRs are the same for both methods.

2.1 Run jDMRgrid on a binned genome

This function uses a grid approach to bin the genome into equal sized bins. User specifies the window and step size as numeric values.

out.dir: PATH to output directory.

window: NUMERIC VALUE specifying bin size.

step: NUMERIC VALUE specifying step size. If bin and step size are equal, we are utilizing non-sliding window approach.

samplefiles: PATH to the text file containing path to samples and sample names.

contexts: VECTOR or CHARACTER presenting sequence contexts of the cytosine. By default this option is set to c("CG", "CHG", "CHH"). If you want to run for a single context such as CG, set it as "CG".

min.C: NUMERIC VALUE specifying percentile threshold based on empirical distribution of the cytosines across bins.

mincov: NUMERIC VALUE specifying minimum read coverage over cytosines. By default this option is set as 0.

include.intermediate: LOGICAL specifying whether or not the intermediate component should be included in the HMM model. By default this option is set as FALSE.

runName: CHARACTER as the name of the operation. By default this option is set to 'GridGenome'.

```
library(jDMRgrid)
setwd('/home/robert/jDMRgrid_test')
out.dir.pop <- "/home/robert/jDMRgrid_test/folder_population"
out.dir.rep <- "/home/robert/jDMRgrid_test/folder_replicate"
samplefile <- system.file("extdata", "listFiles2.fn", package="jDMRgrid")

runjDMRgrid(out.dir = paste0(out.dir.pop, '/grid'),
            window = 200,
            step = 50,
            samplefiles = system.file("extdata", "listFiles1.fn", package="jDMRgrid"),
            contexts = c("CG", "CHG", "CHH"),
            min.C = 10,
            mincov = 0,
            include.intermediate = FALSE,
            runName = "Arabidopsis")

runjDMRgrid(out.dir = paste0(out.dir.rep, '/grid'),
            window = 200,
            step = 50,
            samplefiles = system.file("extdata", "listFiles2.fn", package="jDMRgrid"),
            contexts = c("CG", "CHG", "CHH"),
            min.C = 10,
            mincov = 0,
            include.intermediate = TRUE, #if you want to include intermediate calls as well
            runName = "Arabidopsis")
```

2.1.1 Output files of jDMR Grid approach

Region files containing state calls and methylation levels will be generated for each sample and for each context.

	seqnames	start	end	context	posteriorMax	status	rc.meth.lvl
1:	1	1	200	CG	1.00000	U	0.14026
2:	1	51	250	CG	0.97557	U	0.15794
3:	1	101	300	CG	0.97557	U	0.15794
4:	1	151	350	CG	0.86032	M	0.76287
5:	1	201	400	CG	0.86032	M	0.76287
6:	1	251	450	CG	0.86032	M	0.76287

seqnames, start and end: 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 Step 2: Generate DMR matrix

3.1 Run “makeDMRmatrix”

This function generates a DMR matrix of state calls, rc.meth.lvls and posterior probabilities for all samples in one dataframe.

samplefiles: PATH to file containing description about the samples

input.dir: PATH to directory containing region files.

out.dir: PATH to output directory.

contexts: sequence contexts of the cytosine. By default this option is set to c(“CG”, “CHG”, “CHH”). If you want to run for a single context such as CG, set it as “CG”.

postMax.out: By default this option is set as FALSE. You can set it to TRUE if you want to output the DMR matrix containing posterior probabilities for the status call of each region.

```
makeDMRmatrix(contexts = c("CG", "CHG", "CHH"), postMax.out = TRUE, samplefiles = system.file("extdata",
"listFiles1.fn", package = "jDMRgrid"), input.dir = paste0(out.dir.pop, "/grid"),
out.dir = paste0(out.dir.pop, "/matrix"), include.intermediate = FALSE)

makeDMRmatrix(contexts = c("CG", "CHG", "CHH"), postMax.out = TRUE, samplefiles = system.file("extdata",
"listFiles2.fn", package = "jDMRgrid"), input.dir = paste0(out.dir.rep, "/grid"),
out.dir = paste0(out.dir.rep, "/matrix"), include.intermediate = FALSE)
```

3.2 Output files of DMRmatrix function

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

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

	seqnames	start	end	methylomeA_rep1	methylomeB_rep1	methylomeC_rep1	methylomeD_rep1	methylomeE_rep1
1:	1	1	200	0	1	1	1	1
2:	1	51	250	0	1	1	1	1
3:	1	101	300	0	1	1	1	1
4:	1	151	350	1	1	0	0	1
5:	1	201	400	1	0	0	0	0
6:	1	251	450	1	0	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.pop, "/matrix/CG_rcMethlvl.txt", sep = ""), header = TRUE)
head(rcmethlvls)
```

	seqnames	start	end	methylomeA_rep1	methylomeB_rep1	methylomeC_rep1	methylomeD_rep1	methylomeE_rep1
1:	1	1	200	0.14026	0.85241	0.84005	0.85982	0.84436
2:	1	51	250	0.15794	0.84972	0.50851	0.76422	0.50986
3:	1	101	300	0.15794	0.84972	0.50851	0.76422	0.50986
4:	1	151	350	0.76287	0.84972	0.49221	0.49467	0.50986
5:	1	201	400	0.76287	0.49249	0.49221	0.49467	0.49228
6:	1	251	450	0.76287	0.49249	0.49221	0.49467	0.49228

“CG_postMax.txt” has the following structure. The output matrix contains posterior probabilities for each sample and for the specific region.

```
postMax <- fread(paste0(out.dir.pop, "/matrix/CG_postMax.txt", sep = ""), header = TRUE)
head(postMax)
```

	seqnames	start	end	methylomeA_rep1	methylomeB_rep1	methylomeC_rep1	methylomeD_rep1	methylomeE_rep1
1:	1	1	200	1.00000	1.00000	0.98644	0.99999	0.98248
2:	1	51	250	0.97557	0.99625	0.52280	0.86909	0.52408

3:	1	101	300	0.97557	0.99625	0.52280	0.86909	0.52408
4:	1	151	350	0.86032	0.99625	0.50000	0.50000	0.52408
5:	1	201	400	0.86032	0.50000	0.50000	0.50000	0.49999
6:	1	251	450	0.86032	0.50000	0.50000	0.50000	0.49999

3.3 Split DMR matrix into pairwise groups (only applicable for datasets with control- treatments)

Ignore this step if you are running jDMR on population data without replicates

samplefiles: PATH to file containing description about the samples

input.dir: PATH to directory containing region files.

out.dir: PATH to output directory.

contexts: sequence contexts of the cytosine. By default this option is set to c("CG", "CHG", "CHH"). If you want to run for a single context such as CG, set it as "CG".

postMax.out: by default this option is set to FALSE. If you want to output the matrix containing posterior probabilities set it to TRUE.

```
split.groups(samplefiles = system.file("extdata", "listFiles2.fn", package = "jDMRgrid"),
             input.dir = paste0(out.dir.rep, "/matrix"), out.dir = paste0(out.dir.rep, "/matrix"))
```

4 Step 3: Filter DMR matrix

4.1 Filter the DMR matrix

This function filters the DMR matrix for non-polymorphic patterns.

data.dir: PATH to folder containing DMR matrix

epiMAF.cutoff: Applicable for calling calling population DMRs. This option can be used to filter for Minor Epi-Allele frequency as specified by user. By default, this option is set to NULL.

replicate.consensus : Applicable for control-treatment data-sets with replicates. Users 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.

samplefiles : Path to the text file containing path to samples and sample names. For control/treatment data an additional column specifying the replicates is required.

```
filterDMRmatrix(epiMAF.cutoff = 0.33, replicate.consensus = NULL, data.dir = paste0(out.dir.pop,
"/matrix"), samplefiles = system.file("extdata", "listFiles1.fn", package = "jDMRgrid"))

filterDMRmatrix(epiMAF.cutoff = NULL, replicate.consensus = 0.8, data.dir = paste0(out.dir.rep,
"/matrix"), samplefiles = system.file("extdata", "listFiles2.fn", package = "jDMRgrid"))
```

4.2 Filtered Output

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

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

	seqnames	start	end	methyloA_rep1	methyloB_rep1	methyloC_rep1	methyloD_rep1	methyloE_rep1
1:	1	1	300	0	1	1	1	1
2:	1	151	350	1	1	0	0	1
3:	1	201	500	1	0	0	0	0
4:	1	351	600	0	0	0	1	0
5:	1	451	700	1	0	1	1	0
6:	1	551	750	1	0	1	0	0

If "rc.methlvl.out" option is set to TRUE a filtered matrix with averaged methylation levels is generated.

```
rcmethlvlFiltered <- fread(paste0(out.dir.pop, "/matrix/CG_rcMethlvl-filtered.txt",
sep = ""), header = TRUE)
head(rcmethlvlFiltered)
```

	seqnames	start	end	methyloA_rep1	methyloB_rep1	methyloC_rep1	methyloD_rep1	methyloE_rep1
1:	1	1	300	0.1520467	0.8506167	0.6190233	0.7960867	0.6213600
2:	1	151	350	0.7628700	0.8497200	0.4922100	0.4946700	0.5098600
3:	1	201	500	0.7628700	0.4924900	0.4082933	0.4946700	0.4048833
4:	1	351	600	0.5021100	0.2372300	0.2489600	0.7642200	0.2300900
5:	1	451	700	0.8041150	0.2372300	0.8497400	0.6355350	0.3611850
6:	1	551	750	0.8453600	0.4924900	0.8497400	0.2308000	0.4922800

5 Step 4: Search for context-specific and annotate DMRs

5.1 Output context specific DMRs

Output DMRs specific for contexts i.e CG-only, CHG-only, CHH-only, non-CG and multi-context DMRs using the *StateCalls-filtered.txt files.

```
context.specific.DMRs(samplefiles = system.file("extdata", "listFiles1.fn", package = "jDMRgrid"),
  output.dir = paste0(out.dir.pop, "/context_DMRs"), input.dir = paste0(out.dir.pop,
    "/matrix"))

context.specific.DMRs(samplefiles = system.file("extdata", "listFiles2.fn", package = "jDMRgrid"),
  output.dir = paste0(out.dir.rep, "/context_DMRs"), input.dir = paste0(out.dir.rep,
    "/matrix"))
```

5.2 Annotate DMRs

This function annotates the lists of DMRs. Any file(.txt) containing 3 columns (chr, start, stop) can be annotated using the annotateDMRs function. Please move all files to be annotated to a separate folder and set the full PATH to the “input.dir” option.

gff.files: Multiple gff3 annotation files can be supplied as a vector

annotation: specify annotation categories

input.dir: path to folder containing only files to be annotated. Any file containing 3 columns (chr, start, stop) can be annotated using the annotateDMRs function.

gff3.out: whether to output annotated files in gff3 format

out.dir: path to output folder

In the following example, I will annotate the files generated in section 4.3

```
gff.file_promoters <- "/home/robert/jDMRgrid/jDMRgrid/toyData/TAIR10_promoters.gff3"
gff.file_TE <- "/home/robert/jDMRgrid/jDMRgrid/toyData/TAIR10_TE.gff3"
gff.file_genes <- "/home/robert/jDMRgrid/jDMRgrid/toyData/TAIR10.gene.chr1.gff3"

annotateDMRs(gff.files = c(gff.file_promoters, gff.file_TE, gff.file_genes),
  annotation = c("promoters", "TE", "gene"), #string containing annotation types
  input.dir = paste0(out.dir.pop, "/context_DMRs"),
  gff3.out = FALSE,
  out.dir = paste0(out.dir.pop, '/annotate_DMRs'))

annotateDMRs(gff.files = c(gff.file_promoters, gff.file_TE, gff.file_genes),
  annotation = c("promoters", "TE", "gene"), #string containing annotation types
  input.dir = paste0(out.dir.rep, "/context_DMRs"),
  gff3.out = FALSE,
  out.dir = paste0(out.dir.rep, '/annotate_DMRs'))
```

5.3 Output files after annotation

Mapped files are output in .txt and/or .gff3 format. Additionally, a DMR count table is generated.

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

	sample	total.DMRs	promoters	TE	gene	multiple.overlaps
1:	WT_mutant1_CG-only-DMRs	384	45	3	191	124
2:	WT_mutant1_CHG-only-DMRs	473	38	6	236	171
3:	WT_mutant1_CHH-only-DMRs	473	63	8	218	161

4: WT_mutant1_multi-context-DMRs	192	24	1	98	60
5: WT_mutant1_nonCG-DMRs	350	37	1	184	108

6 R session info

```
sessionInfo()
```

```
R version 4.0.3 (2020-10-10)
```

```
Platform: x86_64-pc-linux-gnu (64-bit)
```

```
Running under: Oracle Linux Server 8.3
```

```
Matrix products: default
```

```
BLAS: /opt/R/4.0.3/lib64/R/lib/libRblas.so
```

```
LAPACK: /opt/R/4.0.3/lib64/R/lib/libRlapack.so
```

```
locale:
```

```
[1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C              LC_TIME=en_US.UTF-8      LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=en_US.UTF-8   LC_MESSAGES=en_US.UTF-8   LC_PAPER=en_US.UTF-8     LC_NAME=C
[9] LC_ADDRESS=C              LC_TELEPHONE=C            LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
```

```
attached base packages:
```

```
[1] parallel stats4 stats graphics grDevices utils datasets methods base
```

```
other attached packages:
```

```
[1] jDMRgrid_0.2.0      R.utils_2.12.2      R.oo_1.25.0          R.methodsS3_1.8.2    GenomicRanges_1.42.
[6] GenomeInfoDb_1.26.7 IRanges_2.24.1      S4Vectors_0.28.1     BiocGenerics_0.36.1  data.table_1.14.8
```

```
loaded via a namespace (and not attached):
```

```
[1] Rcpp_1.0.10          ape_5.7-1            lattice_0.20-41       tidyr_1.3.0
[5] Rsamtools_2.6.0      Biostrings_2.58.0    digest_0.6.31         utf8_1.2.3
[9] R6_2.5.1             plyr_1.8.8           evaluate_0.21         ggplot2_3.4.2
[13] pillar_1.9.0         zlibbioc_1.36.0      rlang_1.1.1           rstudioapi_0.14
[17] Matrix_1.5-4.1       rmarkdown_2.23       BiocParallel_1.24.1   stringr_1.5.0
[21] RCurl_1.98-1.12      munsell_0.5.0        DelayedArray_0.16.3   compiler_4.0.3
[25] rtracklayer_1.50.0   xfun_0.39            pkgconfig_2.0.3       htmltools_0.5.5
[29] tidyselect_1.2.0     SummarizedExperiment_1.20.0 tibble_3.2.1          GenomeInfoDbData_1.
[33] matrixStats_1.0.0    XML_3.99-0.14        fansi_1.0.4           crayon_1.5.2
[37] dplyr_1.1.2          GenomicAlignments_1.26.0 bitops_1.0-7          grid_4.0.3
[41] nlme_3.1-149         gtable_0.3.3         lifecycle_1.0.3       magrittr_2.0.3
[45] formatR_1.14         scales_1.2.1         cli_3.6.1             stringi_1.7.12
[49] XVector_0.30.0       reshape2_1.4.4       generics_0.1.3        vctrs_0.6.3
[53] methimpute_1.12.0    tools_4.0.3          Biobase_2.50.0        glue_1.6.2
[57] purrr_1.0.1          MatrixGenerics_1.2.1 fastmap_1.1.1         yaml_2.3.7
[61] colorspace_2.1-0     minpack.lm_1.2-3     knitr_1.43
```