

jDMR: a heuristic DMR caller for WGBS data

Rashmi Hazarika, Y.Shahryary & Frank Johannes

2022-02-22

Contents

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

1 Input files

For generation of region-level calls, jDMR 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 = "jDMR")
fread(samplefile2, header = TRUE)
```

	file	sample	replicate	group
1:	methimpute-out/methylome_A_All.txt	WT	rep1	control
2:	methimpute-out/methylome_B_All.txt	WT	rep2	control
3:	methimpute-out/methylome_C_All.txt	mutant1	rep1	treatment1
4:	methimpute-out/methylome_D_All.txt	mutant1	rep2	treatment1
5:	methimpute-out/methylome_E_All.txt	mutant2	rep1	treatment2
6:	methimpute-out/methylome_F_All.txt	mutant2	rep2	treatment2

2 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 jDMR on cytosine clusters extracted from genome

This function starts by identifying cytosine clusters in the genome and uses them in each sample to identify DMRs.

fasta.file: PATH to FASTA file

samplefiles: PATH to file containing description about the samples

genome: a string containing name of the genome

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".

```
library(jDMR)
```

```
out.dir <- "/myfolder/DMR-results"
myfasta <- system.file("extdata", "TAIR10_chr_all.fa", package = "jDMR")
samplefile <- system.file("extdata", "listFiles2.fn", package = "jDMR")

runjDMRregions(fasta.file = myfasta, samplefiles = samplefile, genome = "Arabidopsis",
  out.dir = out.dir)
```

2.1.1 Output files of jDMR Regions approach

Rdata files containing coordinates of Cytosine clusters will be generated for each chromosome and cytosine context.

Output file "Arabidopsis_regions_chr1_CG.Rdata" contains coordinates of cytosine clusters

```
regionfile <- dget(system.file("extdata", "min.C_5/fp0.01/Arabidopsis_regions_chr1_CG.Rdata",
  package = "jDMR"))
```

```
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

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

	seqnames	start	end	context	posteriorMax	status	rc.meth.lvl
1:	1	3696	3856	CG	0.99999	U	0.01730
2:	1	12100	12155	CG	0.99999	U	0.01730
3:	1	20991	21026	CG	0.99999	U	0.01730
4:	1	21257	21293	CG	0.99999	U	0.01730
5:	1	29966	30008	CG	0.99999	M	0.87391
6:	1	46099	46141	CG	0.99999	U	0.01730

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

2.2 Run jDMR on a binned genome

This function uses a grid approach to bin the genome into equal sized bins ranging from 100 to 1000 bps. The optimum bin and step size will be determined automatically for each cytosine context using the min.C parameter.

fasta.file: PATH to FASTA file

samplefiles: PATH to file containing description about the samples

genome: a string containing name of the genome

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".

min.C: minimum number of cytosines in a bin. bins lower than specified threshold will be dropped.

```
library(jDMR)

out.dir <- "/myfolder/DMR-results"
myfasta <- system.file("extdata", "TAIR10_chr_all.fa", package = "jDMR")
samplefile <- system.file("extdata", "listFiles2.fn", package = "jDMR")

runjDMRgrid(out.dir = out.dir, fasta.file = myfasta, samplefiles = samplefile, min.C = 10,
            genome = "Arabidopsis")
```

2.2.1 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	500	CG	0.99999	M	0.74660
2:	1	501	1000	CG	0.99999	M	0.74660
3:	1	1001	1500	CG	0.99999	U	0.03057
4:	1	1501	2000	CG	0.99999	U	0.03057
5:	1	2001	2500	CG	0.99999	U	0.03057
6:	1	2501	3000	CG	0.99999	U	0.03057

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 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.

```
input.dir <- "/myfolder/DMR-results"
out.dir <- "/myfolder/DMRmatrix-results"
samplefile <- system.file("extdata", "listFiles2.fn", package = "jDMR")

makeDMRmatrix(samplefiles = samplefile, input.dir = myinput, out.dir = out.dir)
```

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, "CG_StateCalls.txt", sep = ""), header = TRUE)
head(statecalls)
```

	seqnames	start	end	WT_rep1	WT_rep2	mutant1_rep1	mutant1_rep2	mutant2_rep1	mutant2_rep2
1:	1	1	500	1	1	1	1	1	1
2:	1	501	1000	1	1	1	1	1	1
3:	1	1001	1500	0	0	0	0	0	0
4:	1	1501	2000	0	0	0	0	0	0
5:	1	2001	2500	0	0	0	0	0	0
6:	1	2501	3000	0	0	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, "CG_rcMethlvl.txt", sep = ""), header = TRUE)
head(rcmethlvls)
```

	seqnames	start	end	WT_rep1	WT_rep2	mutant1_rep1	mutant1_rep2	mutant2_rep1	mutant2_rep2
1:	1	1	500	0.74660	0.74660	0.67491	0.71750	0.71563	0.67364
2:	1	501	1000	0.74660	0.74660	0.67491	0.71750	0.71563	0.67364
3:	1	1001	1500	0.03057	0.03057	0.01907	0.02158	0.02159	0.01996
4:	1	1501	2000	0.03057	0.03057	0.01907	0.02158	0.02159	0.01996
5:	1	2001	2500	0.03057	0.03057	0.01907	0.02158	0.02159	0.01996
6:	1	2501	3000	0.03057	0.03057	0.01907	0.02158	0.02159	0.01996

“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, "CG_postMax.txt", sep = ""), header = TRUE)
head(postMax)
```

	seqnames	start	end	WT_rep1	WT_rep2	mutant1_rep1	mutant1_rep2	mutant2_rep1	mutant2_rep2
1:	1	1	500	0.99999	0.99999	1	0.99999	0.99999	1
2:	1	501	1000	0.99999	0.99999	1	0.99999	0.99999	1
3:	1	1001	1500	0.99999	0.99999	1	0.99999	0.99999	1
4:	1	1501	2000	0.99999	0.99999	1	0.99999	0.99999	1
5:	1	2001	2500	0.99999	0.99999	1	0.99999	0.99999	1
6:	1	2501	3000	0.99999	0.99999	1	0.99999	0.99999	1

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.

```
samplefile <- system.file("extdata", "listFiles2.fn", package = "jDMR")

split.groups(samplefiles = samplefile, input.dir = "/myfolder/DMRmatrix-results",
  out.dir = "/myfolder/DMRmatrix-results/split_gps")
```

4 Filter DMR matrix

4.1 Filter the DMR matrix

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

gridDMR: set this option to TRUE if Grid approach was used otherwise set to FALSE. The output will contain merged regions.

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.

rc.methlvl.out: Output filtered matrix containing recalibrated methylation levels. By default, this option is set to FALSE.

```
out.dir <- "/myfolder/DMRmatrix-results/split_gps"
filterDMRmatrix(gridDMR=TRUE, # setting to TRUE because we are using the outputs of grid approach
  data.dir=out.dir,
  replicate.consensus=1) #since we have 2 replicates for each sample
```

4.2 Filtered Output

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

```
statecallsFiltered <- fread(paste0(my.dir, "CG_WT_mutant1_StateCalls-filtered.txt",
  sep = ""), header = TRUE)
head(statecallsFiltered)
```

	seqnames	start	end	width	WT_rep1	WT_rep2	mutant1_rep1	mutant1_rep2
1:	1	32001	32500	500	0	0	1	1
2:	1	44501	45000	500	1	1	0	0
3:	1	95001	95500	500	0	0	1	1
4:	1	133001	133500	500	1	1	0	0
5:	1	240501	241000	500	0	0	1	1
6:	1	290501	291000	500	0	0	1	1

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

```
rcmethlvlFiltered <- fread(paste0(my.dir, "CG_WT_mutant1_rcmethlvl-filtered.txt",
  sep = ""), header = TRUE)
head(rcmethlvlFiltered)
```

	seqnames	start	end	width	WT_rep1	WT_rep2	mutant1_rep1	mutant1_rep2
1:	1	32001	32500	500	0.03056	0.03056	0.52215	0.71750
2:	1	44501	45000	500	0.74660	0.74660	0.01906	0.02157
3:	1	95001	95500	500	0.03056	0.03056	0.67490	0.71750
4:	1	133001	133500	500	0.74658	0.74658	0.01906	0.02157
5:	1	240501	241000	500	0.03056	0.03056	0.67490	0.71750
6:	1	290501	291000	500	0.03056	0.03056	0.67490	0.71750

4.3 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.

```
samplefile <- system.file("extdata", "listFiles2.fn", package = "jDMR")
out.dir <- "/myfolder/DMRmatrix-results/split_gps"

context.specific.DMRs(samplefiles = samplefile, data.dir = out.dir)
```

5 Annotate DMRs

This function annotates the lists of DMRs.

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

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

mydir <- "/myfolder/annotate_DMRs/"

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 after annotation

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

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

	sample	total.DMRs	gene	promoters	TE	multiple.overlaps
1:	WT_mutant1_CG-only-DMRs	269	157	13	15	59
2:	WT_mutant1_CHG-only-DMRs	193	43	15	62	30

3: WT_mutant1_CHH-only-DMRs	48364	21444	3423	9746	8887
4: WT_mutant1_nonCG-DMRs	7	3	0	2	1
5: WT_mutant2_CG-only-DMRs	261	145	15	18	64
6: WT_mutant2_CHG-only-DMRs	135	19	14	53	22
7: WT_mutant2_CHH-only-DMRs	44037	21015	3304	6936	8612
8: WT_mutant2_nonCG-DMRs	4	2	0	1	1

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] jDMR_0.1.0 R.utils_2.11.0 R.oo_1.24.0 R.methodsS3_1.8.1 GenomicRanges_1.42.0
[6] GenomeInfoDb_1.26.7 IRanges_2.24.1 S4Vectors_0.28.1 BiocGenerics_0.36.1 data.table_1.14.2

loaded via a namespace (and not attached):
[1] Rcpp_1.0.8 lattice_0.20-45 tidyr_1.2.0 Rsamtools_2.6.0
[5] Biostrings_2.58.0 assertthat_0.2.1 digest_0.6.29 utf8_1.2.2
[9] R6_2.5.1 plyr_1.8.6 evaluate_0.14 ggplot2_3.3.5
[13] pillar_1.7.0 zlibbioc_1.36.0 rlang_1.0.1 rstudioapi_0.13
[17] Matrix_1.4-0 rmarkdown_2.11 BiocParallel_1.24.1 stringr_1.4.0
[21] RCurl_1.98-1.5 munsell_0.5.0 DelayedArray_0.16.3 compiler_4.0.1
[25] rtracklayer_1.50.0 xfun_0.29 pkgconfig_2.0.3 htmltools_0.5.2
[29] tidyselect_1.1.2 SummarizedExperiment_1.20.0 tibble_3.1.6 GenomeInfoDbData_1.2.6
[33] matrixStats_0.61.0 XML_3.99-0.8 fansi_1.0.2 crayon_1.5.0
[37] dplyr_1.0.8 GenomicAlignments_1.26.0 bitops_1.0-7 grid_4.0.1
[41] gtable_0.3.0 lifecycle_1.0.1 DBI_1.1.2 magrittr_2.0.2
[45] formatR_1.11 scales_1.1.1 cli_3.2.0 stringi_1.7.6
[49] XVector_0.30.0 reshape2_1.4.4 ellipsis_0.3.2 generics_0.1.2
[53] vctrs_0.3.8 methimpute_1.12.0 tools_4.0.1 Biobase_2.50.0
[57] glue_1.6.1 purrr_0.3.4 MatrixGenerics_1.2.1 fastmap_1.1.0
[61] yaml_2.2.2 colorspace_2.0-3 minpack.lm_1.2-1 knitr_1.37