MethylSeekR

Lukas Burger, Dimos Gaidatzis, Dirk Schübeler and Michael Stadler

Modified: July 25, 2013. Compiled: July 26, 2013

Contents

1 Introduction

This vignette describes how to use *MethylSeekR* to find regulatory regions from whole-genome bisulfite-sequencing (Bis-seq) methylation data. The basic ideas of the method were introduced in [1] and the validity of the approach demonstrated on Bis-seq data from mouse embryonic stem cells and neural progenitors. MethylSeekR represents a refined and extended version of the inital approach that makes it more robust and generally applicable and allows much simpler and interactive setting of segmentation parameters. For details regarding the methodology, we refer the reader to [2]. If you use MethylSeekR in a publication, please cite [2].

2 Prerequisites

Several functions of *MethylSeekR* require the genome sequence or derived information such as the chromsome lengths of the reference genome used. To retrieve this information, *MethylSeekR* makes use of the *BSgenome* package and the related genome data packages. *BSgenome* can be installed from Bioconductor (`http://www.bioconductor.org`) using the following commands

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("BSgenome")
```

After the installation, the available genomes can be displayed with

```
> library(BSgenome)
> available.genomes()
```

For the example data that comes along with the package, we need the hg18 assembly of the human genome *BSgenome.Hsapiens.UCSC.hg18*. This genome, and analogously any of the other genomes, can be installed as follows

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("BSgenome.Hsapiens.UCSC.hg18")
```

3 Data import and SNP filtering

We start by loading the *MethylSeekR* package

```
> library(MethylSeekR)
```

One step of the *MethylSeekR* workflow depends on random permutations of the data (see below), which may lead to slightly different results each time the calculations are performed. It is thus recommended to set the random number generator seed at the beginning of the analysis to ensure exact reproducibility of the results. In R, this can be done e.g. as follows

```
> set.seed(123)
```

In a first step, the Bis-seq data needs to be loaded into R. The data should be in a tab-delimited file with four columns, *chromosome*, *position*, T and M, where *position* is the coordinate of the cytosine in the CpG on the plus strand (we assume that the counts have been pooled from the two strands), T is the total number of reads overlapping the cytosine and M the total number of reads without cytosine to thymine conversion (reflecting methylated cytosines). An example file, containing the methylation information for $200,000$ CpGs from chromosome 22 of human IMR90 cells [3], is part of the package and its path can be retrieved with the following command

```
> system.file("extdata", "Lister2009_imr90_hg18_chr22.tab",
+ package="MethylSeekR")
```

The data can be loaded into R using the `readMethylome` function. This function takes two arguments, the name of the file containing the methylation data and a named vector containing the chromosome lengths of the reference genome. This vector can be easily created using the *BSgenome* package of the reference genome. For the example data used here, we work with the human hg18 assembly.

```
> library("BSgenome.Hsapiens.UCSC.hg18")
> sLengths=seqlengths(Hsapiens)
> head(sLengths)
     chr1      chr2      chr3      chr4      chr5      chr6
247249719 242951149 199501827 191273063 180857866 170899992
```

`readMethylome` will directly create a GRanges object containing the methylation data. GRanges objects (as defined in the *GenomicRanges* package) represent general purpose containers for storing genomic intervals and allow for very efficient overlap/intersection operations. To use *MethylSeekR*, it is not necessary to understand how GRanges objects work, and we refer the interested reader to `help(package="GenomicRanges")`.

To read in the methylation data of our example data, we can use the following commands.

```
> methFname <- system.file("extdata",
+ "Lister2009_imr90_hg18_chr22.tab", package="MethylSeekR")
> meth.gr <- readMethylome(FileName=methFname, seqLengths=sLengths)
> head(meth.gr)
GRanges with 6 ranges and 2 metadata columns:
      seqnames               ranges strand |         T         M
         <Rle>            <IRanges>  <Rle> | <numeric> <numeric>
  [1]    chr22 [21971222, 21971222]      * |        20        19
  [2]    chr22 [21971252, 21971252]      * |         8         8
  [3]    chr22 [21971269, 21971269]      * |         3         3
  [4]    chr22 [21971308, 21971308]      * |         1         1
  [5]    chr22 [21971316, 21971316]      * |         3         3
  [6]    chr22 [21971328, 21971328]      * |         4         4
  ---
  seqlengths:
           chr1          chr2          chr3 ...  chr22_random   chrX_random
      247249719     242951149     199501827 ...        257318       1719168
```

Each element of the `meth.gr` object stands for one CpG and the corresponding metadata (columns on the right) contains its counts T and M.

In cases where the genomic background of the sample differs from the reference genome which the methylation data has been aligned against, CpGs overlapping SNPs should be removed from the methylation table [1, 2]. To this end, `MethylSeekR` provides functions to both load a SNP table and remove the SNP-overlapping CpGs from the methylation GRanges object (the following four code chunks can be skipped if the genomic background of the sample is identical to the reference genome).

Similary to the methylation data, the SNP table can be loaded using the `readSNPTable` function. The SNP data should be in a tab-delimited text file with two columns corresponding to the chromosome and position of the SNPs. The path to an example SNP file with data downloaded from dbSNP [4] for hg18 (and prefiltered to keep the file size small) can be retrieved as follows

```
> system.file("extdata", "SNVs_hg18_chr22.tab",
+ package="MethylSeekR")
```

To read in the SNPs from our example file, we can thus use the following commands

```
> snpFname <- system.file("extdata", "SNVs_hg18_chr22.tab",
+ package="MethylSeekR")
> snps.gr <- readSNPTable(FileName=snpFname, seqLengths=sLengths)
> head(snps.gr)
GRanges with 6 ranges and 0 metadata columns:
      seqnames               ranges strand
         <Rle>            <IRanges>  <Rle>
  [1]    chr22 [21971332, 21971332]      *
  [2]    chr22 [21971450, 21971450]      *
  [3]    chr22 [21974271, 21974271]      *
  [4]    chr22 [21974631, 21974631]      *
  [5]    chr22 [21974794, 21974794]      *
  [6]    chr22 [21975493, 21975493]      *
  ---
  seqlengths:
           chr1          chr2          chr3 ...  chr22_random   chrX_random
      247249719     242951149     199501827 ...        257318       1719168
```

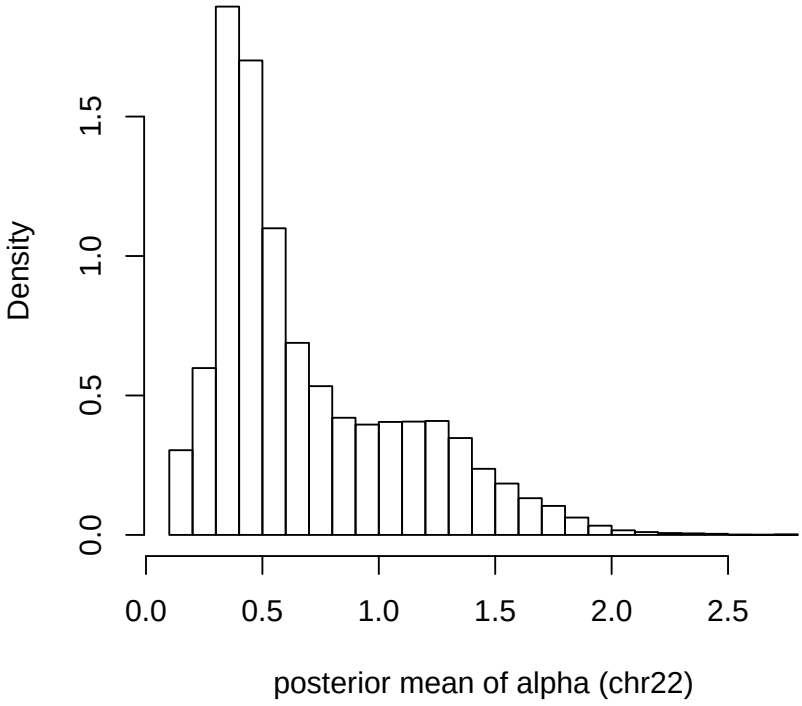
Here, `seqLengths` is the same named vector with chromsome lengths as needed for the `readMethylome` function. `readSNPTable` directly returns a GRanges object containing the coordinates of the SNPs. Using this object, the CpGs overlapping SNPs can now be removed from the methylation object using the `removeSNPs` function.

```
> meth.gr <- removeSNPs(meth.gr, snps.gr)
```

4 Segmentation of partially methylated domains

Some methylomes contain large regions of seemingly disordered methylation which have been termed partially methylated domains (PMDs, [3]). These regions need to be identified and masked prior to the segmentation of regulatory regions. To decide whether a methylome contains PMDs, we can first calculate the distribution of α-values for one selected chromosome. α characterizes the distribution of methylation levels in sliding windows containing 100 consecutive CpGs along the genome. α-values smaller than 1 reflect a polarized distribution, favoring low and high methylation (as in regulatory regions and background methylation levels respectively), whereas α-values equal to 1 or larger indicate distributions that are rather uniform or polarized towards intermediate methylation levels (as in PMDs). The distribution of α-values for one selected chromosome can be determined as follows,

```
> plotAlphaDistributionOneChr(m=meth.gr, chr.sel="chr22",
+ num.cores=1)
```



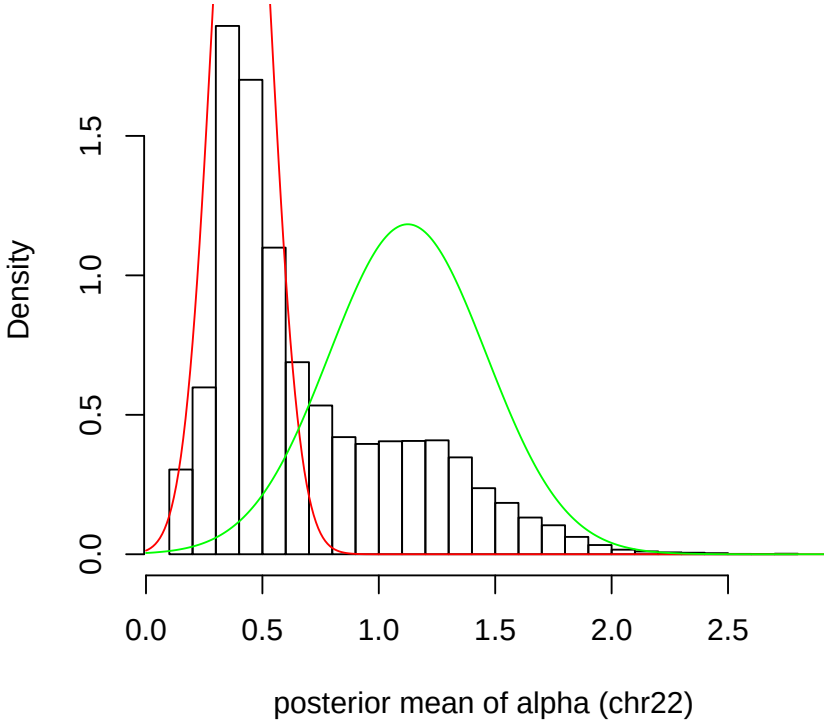
`chr.sel` denotes the chromosome for which the statistic is calculated. Generally, the α distribution of different chromosomes are very similar, but we recommend using a small autosome to minimize the computing time and to avoid sex-specific biases. `plotAlphaDistributionOneChr` generates a figure showing the inferred α distribution. This figure is by default printed to the screen but can also be saved in a pdf file if a filename is provided via the `pdfFilename` argument. `num.cores` is the number of cores that are used for the calculation. By default, all functions of *MethylSeekR* perform their calculations on a single core. However, almost all functions can be run in parallel (by setting `num.cores` to a number larger than one, only availabe on unix and mac) and we recommend the user to use multiple cores in order to speed up the calculations. To determine the number of cores of the machine used, use the following commands

```
> library(parallel)
> detectCores()
```

Generally, there is evidence for PMDs if the distribution of α-values is bimodal or long-tailed with a significant fraction of α values larger or equal 1. In the example here, the distribution is clearly bimodal and we thus, in a next step, run a Hidden Markov Model (HMM) to identify PMDs genome-wide (in cases where the α distribution is unimodal (for examples see [2]), there is no evidence for PMDs and the following three code chunks can be omitted).

```
> PMDsegments.gr <- segmentPMDs(m=meth.gr, chr.sel="chr22",
+ seqLengths=sLengths, num.cores=1)
> head(PMDsegments.gr)
GRanges with 6 ranges and 2 metadata columns:
      seqnames               ranges strand |        type       nCG
```

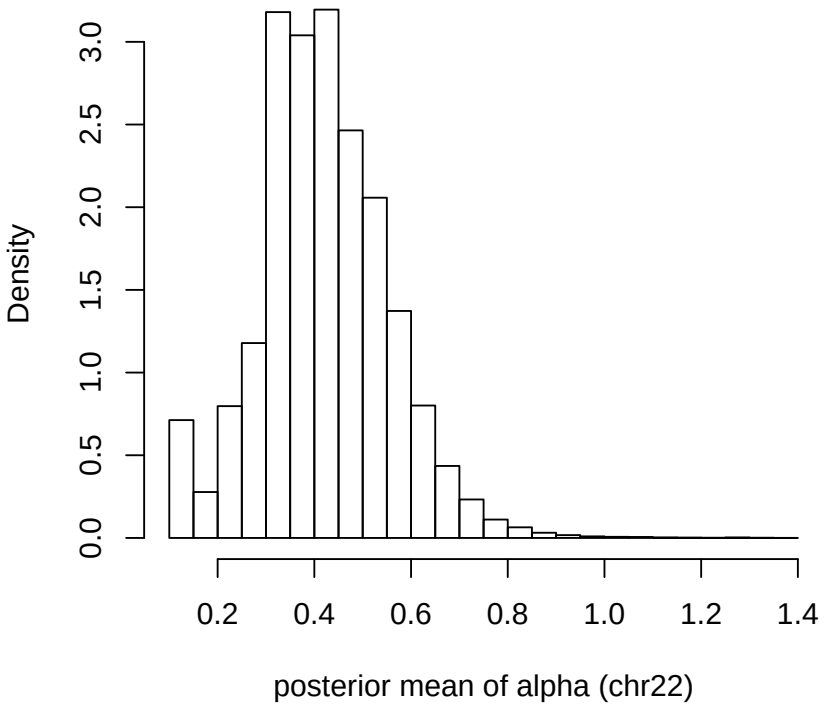
```
      <Rle>            <IRanges>  <Rle> | <character> <integer>
[1]    chr22 [       1, 21992683]     * |      notPMD       337
[2]    chr22 [21992684, 22188740]     * |         PMD      2877
[3]    chr22 [22188741, 22195029]     * |      notPMD       208
[4]    chr22 [22195030, 22236183]     * |         PMD       569
[5]    chr22 [22236184, 22240087]     * |      notPMD       146
[6]    chr22 [22240088, 22304990]     * |         PMD       730
---
seqlengths:
    chr22
 49691432
```



Here, as above, `chr.sel` is the chromosome used to train the HMM. `segmentPMDs` generates a GRanges object which partitions the genome into PMDs and regions outside of PMDs (notPMD) as indicated in the first column of the metadata (`type`). The second column (`nCG`) of the metadata indicates the number of covered (by at least one read) CpGs in each region. The function creates a figure (which is by default printed to the screen or saved in a pdf file if a filename is provided via the `pdfFilename` argument) which shows the same alpha distribution as in the previous figure, but including the fitted Gaussian emission distributions of the HMM in red and green. If the HMM fitting works properly, the two Gaussians should correspond to the two peaks of the bimodal distribution (or the main peak and the tail if the alpha distribution is long-tailed), as illustrated for the example dataset. As a control, we may want to recalculate the distribution of α-values using only the methylation values outside of the predicted PMDs.

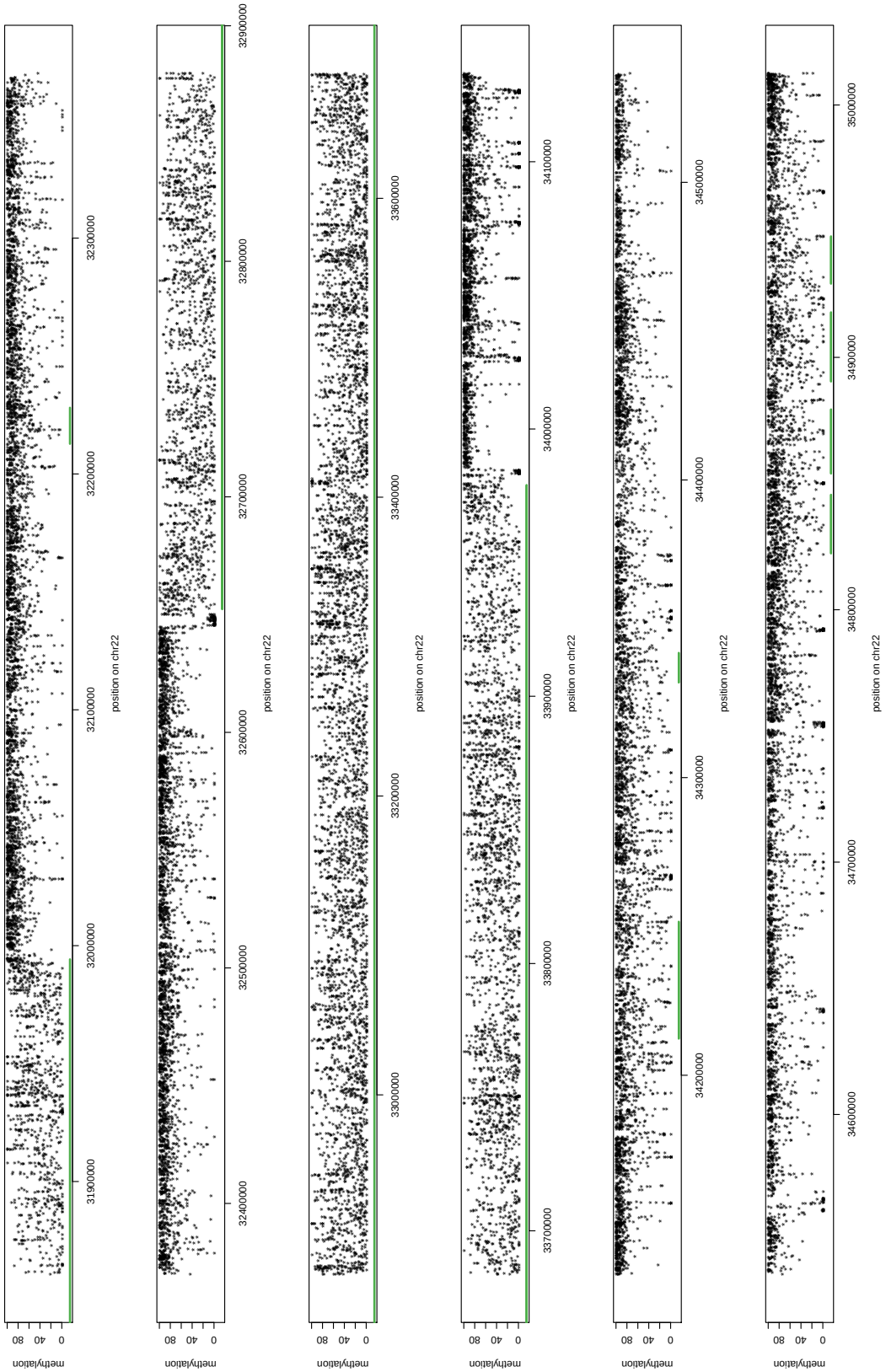
```
> plotAlphaDistributionOneChr(m=subsetByOverlaps(meth.gr,
+ PMDsegments.gr[values(PMDsegments.gr)$type=="notPMD"]), chr.sel="chr22",
```

```
+ num.cores=1)
```



Clearly, we have managed to remove the second mode of the distribution. Such a unimodal distribution with most α-values below 1 is also what is typically seen in methylomes without PMDs. Note that the distribution of α-values is not only influenced by the presence of PMDs, but also by the variability in background methylation levels outside of regulatory regions. In methylomes that do not contain PMDs, but show large variability, a substantial fraction of α-values (10-20%) may be larger than one. However, if there are no PMDs, the shape of the α-distribution should still be unimodal. It is thus rather the shape of the distribution (bimodal versus unimodal) that indicates the presence or absence of PMDs and not the fraction of α-values larger 1. In any case, the resulting PMD segmentation should be visually inspected to assess its validity. This can easily be done for randomly chosen regions of the genome using the `plotPMDSegmentation` function.

```
> plotPMDSegmentation(m=meth.gr, segs=PMDsegments.gr)
```



The resulting figure is either printed to the screen (default) or saved in a pdf file if a filename is provided via the `pdfFilename` argument. It is also possible to plot multiple regions and save them

in a multi-page pdf file if both the argument `pdfFilename` and `numRegions` are provided. The randomly chosen region that is displayed is broken up into 6 panels and in each panel, the raw (ie unsmoothed) methylation levels of all CpGs with a minimal coverage of 5 reads are shown. PMDs are indicated as green bars, extending over the entire PMD.

The PMD segmentation can be saved both as a GRanges object and as a tab-delimited file.

```
> savePMDSegments(PMDs=PMDsegments.gr,
+ GRangesFilename="PMDs.gr.rds", TableFilename="PMDs.tab")
```

`GRangesFilename` is the filename for the GRanges object and the `TableFilename` the filename for the tab-delimited file (only of the two filenames is required).

5 Identification of UMRs and LMRs

Once we have identified PMDs (if there are any), we can proceed to the segmentation of regulatory regions. These come in two distinct groups, unmethylated regions (UMRs) which correspond mostly to unmethylated CpG islands (including promoters) and low-methylated regions (LMRs) which correspond to distal regulatory regions [1, 2]. We identify these regions by a fairly straightforward procedure. We first smooth methylation levels over 3 consecutive CpGs to reduce the sampling noise and then identify hypo-methylated regions as regions of consecutive CpGs with smoothed methylation levels below a fixed cut-off m, containing a minimal number of CpGs n. The identified regions are then further classified into UMRs and LMRs based on the number of CpGs they contain. The parameters m and n must be determined prior to segmentation, which can be done via false discovery rate (FDR) considerations. This approach will naturally take into account the variability of methylation levels in the methylome studied. Briefly, for each set of parameters (m, n), we determine the number of regions with methylation levels below m and with a minimal number of CpGs n in both the original methylome, n_o, and a randomized methylome, n_r. The FDR is then defined as the ratio $\frac{n_r}{n_o}$. Given a user-defined cut-off on the FDR, we can then determine reasonable sets of values for m and n.

To calculate FDRs with `MethylSeekR`, we can use the function `calculateFDRs`. Since the FDR calculation is based on the randomization of methylation levels outside of CpG islands [2], we first need to create a GRanges object containing the CpG islands, which is then passed on to `calculateFDRs`. CpG island annotation can for example be retrieved from UCSC using the package `rtracklayer`

```
> library(rtracklayer)
> session <- browserSession()
> genome(session) <- "hg18"
> query <- ucscTableQuery(session, "cpgIslandExt")
> CpGislands.gr <- track(query, asRangedData = FALSE)
> genome(CpGislands.gr) <- NA
```

If other genome assemblies than hg18 are used, hg18 in the code above needs to be replaced by the corresponding assembly.

Next, we extend the CpG islands to 5kb to make sure that all unmethylated CpGs lying in CpG islands are excluded from the FDR calculation.

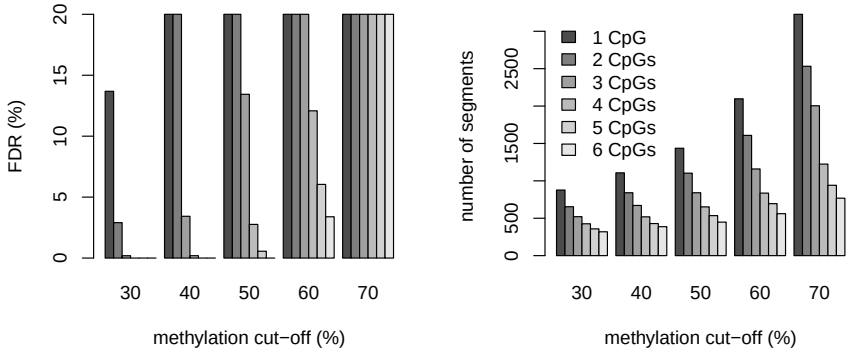
```
> CpGislands.gr <-
+ suppressWarnings(resize(CpGislands.gr, 5000, fix="center"))
```

We can now calculate the FDRs

```
> stats <- calculateFDRs(m=meth.gr, CGIs=CpGislands.gr,
+ PMDs=PMDsegments.gr, num.cores=1)
> stats
```

```
$FDRs
            1          2           3          4          5         6
0.3  13.68301   2.905199   0.1919386  0.0000000  0.0000000  0.000000
0.4  40.92141  20.451843   3.4277198  0.1926782  0.0000000  0.000000
0.5  75.83565  42.429737  13.4363853  2.7607362  0.5607477  0.000000
0.6 136.43300  95.335821  54.3572045 12.0813397  6.0431655  3.386809
0.7 163.71243 158.609795 161.9261477 54.4934641 38.2571732 24.088542

$numSegments
       1    2    3    4   5   6
0.3  877  654  521  426 358 319
0.4 1107  841  671  519 429 387
0.5 1436 1103  841  652 535 448
0.6 2097 1608 1159  836 695 561
0.7 3227 2532 2004 1224 941 768
```



Since the FDR calculation also masks PMDs, the PMDs need to be passed to the function as well. In the case of methylomes without PMDs, the `PMDs` argument can be omitted. The function creates a figure (which is, again, by default printed to the screen or saved in a pdf file if a filename is provided via the `pdfFilename` argument) which illustrates the relationship between FDR (cut at 20% if larger than 20%), m, n and the number of identified segments. Additionally, the function returns a list containing a matrix with FDR values (in percent) and a matrix with the number of inferred segments in the original methylome for each m (rows) and n (columns). Since the parameters m and n are partially redundant, we need to set one parameter by hand. $m = 0.5$ (50% methylation) appears to be a reasonable choice for most methylomes (leading to high accuracy at a good sensitivity when overlapping the identified regions with DNase hypersensitive sites [1, 2]) and we thus set $m = 0.5$ and determine n by choosing the smallest n such that the FDR is below a given cut-off (here 5%).

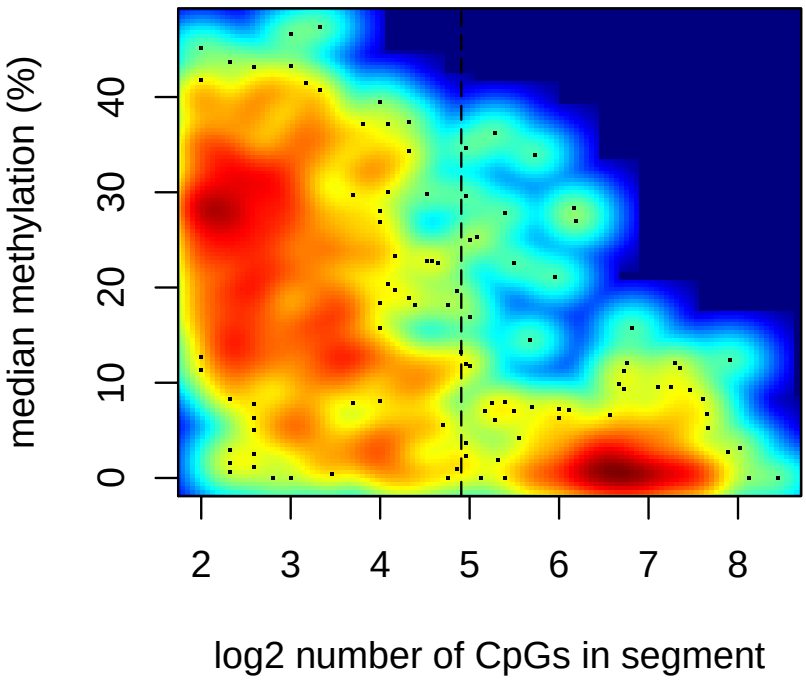
```
> FDR.cutoff <- 5
> m.sel <- 0.5
> n.sel=as.integer(names(stats$FDRs[as.character(m.sel), ]
+ [stats$FDRs[as.character(m.sel), ]<FDR.cutoff])[1])
> n.sel
```

```
[1] 4
```

Once the parameters m (`m.sel`) and n (`n.sel`) have been set, we can call `segmentUMRsLMRs` to identify UMRs and LMRs.

```
> UMRLMRsegments.gr <- segmentUMRsLMRs(m=meth.gr, meth.cutoff=m.sel,
+ nCpG.cutoff=n.sel, PMDs=PMDsegments.gr,
+ num.cores=1, myGenomeSeq=Hsapiens,
+ seqLengths=sLengths)
> head(UMRLMRsegments.gr)
```

```
GRanges with 6 ranges and 7 metadata columns:
      seqnames               ranges strand | nCG.segmentation       nCG       T
         <Rle>            <IRanges>  <Rle> |          <array> <integer> <array>
  [1]    chr22 [21977405, 21978426]      * |               18        19     345
  [2]    chr22 [21982822, 21983618]      * |                6        20     106
  [3]    chr22 [21987770, 21988330]      * |               10        12     254
  [4]    chr22 [22192943, 22195260]      * |               89       152    1288
  [5]    chr22 [22237330, 22239198]      * |               83       109     957
  [6]    chr22 [22388745, 22389856]      * |               41        88     579
           M     pmeth median.meth        type
    <array>   <array>   <numeric> <character>
  [1]      48 0.1391304  0.11594203         LMR
  [2]      38 0.3584906  0.37350218         LMR
  [3]      38 0.1496063  0.16949495         LMR
  [4]     210 0.1630435  0.09523810         UMR
  [5]      78 0.0815047  0.02722222         UMR
  [6]      83 0.1433506  0.01754386         UMR
  ---
  seqlengths:
           chr1          chr2          chr3 ...  chr22_random   chrX_random
      247249719     242951149     199501827 ...        257318       1719168
```



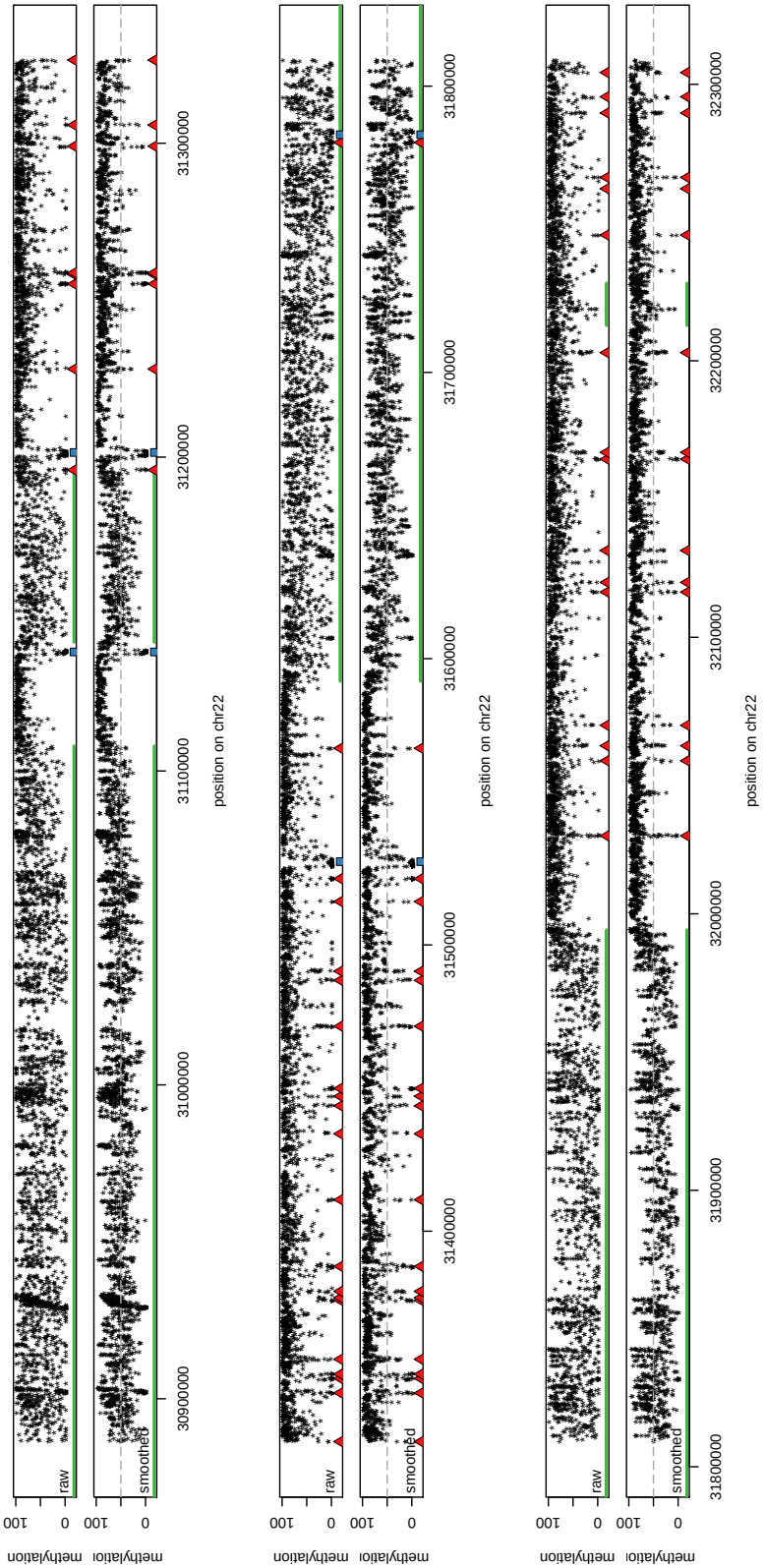
If the methylome under consideration contains no PMDs, the `PMDs` argument can be omitted. The argument `myGenomeSeq` takes as input a BSgenome object of the genome assembly used, here Hsapiens, as defined in the package `BSgenome.Hsapiens.UCSC.hg18` (the genome object is directly loaded with the package and corresponds to the species name in the package name). The genome sequence is needed to retrieve the number of CpGs in the genome per identified region, which is further used to classifiy the regions into UMRs and LMRs. `segmentUMRsLMRs` returns a GRanges object containing the identified regions. The metadata includes the number of covered CpGs (by at least 5 reads) per region (*nCG.segmentation*) , the number of CpGs in the genome per region (*nCG*), the total number of reads overlapping CpGs (T), the total number of tags overlapping CpGs without conversion event (M), the mean methylation ($pmeth = \frac{M}{T}$), the median methylation (*median.meth*, median over the methylation levels of each single CpG) and *type*, which classifies the region into UMR or LMR. For the calculation of T, M, *pmeth* and *median.meth*, only CpG that are covered by at least 5 reads are used. `segmentUMRsLMRs` also creates a figure (by default printed to the screen or saved in a pdf file if a filename is provided via the `pdfFilename` argument) which shows the number of CpGs per region versus its median methylation level. Typically, as in the example here, this figure should show a clear separation into unmethylated, CpG-rich regions (UMRs) versus low-methylated, CpG-poor regions (LMRs). The dashed line is at 30 CpGs and represents the cut-off used for the classification.

Analogously to the PMD segmentation, the result of this final segmentation can be visualized for a randomly selected region.

```
> plotFinalSegmentation(m=meth.gr, segs=UMRLMRsegments.gr,
+ PMDs=PMDsegments.gr,meth.cutoff=m.sel)
```

If the methylome under study does not contain any PMDs, the `PMDs` argument can be omitted.

The resulting figure is either printed to the screen (default) or saved in a pdf file if a filename is provided via the `pdfFilename` argument. It is also possible to plot multiple regions and save them in a multi-page pdf file if both the argument `pdfFilename` and `numRegions` are provided. The randomly chosen region that is displayed is broken up into 3 pairs of panels, where in each pair the same region is shown twice, once with raw methylation levels (top) and once with methylation levels smoothed over 3 consecutive CpGs (bottom). In both cases only CpGs with a coverage of at least 5 reads are shown. The raw data best illustrates the disordered nature of methylation levels in PMDs, whereas the smoothed methylation levels more clearly show UMRs and LMRs. In all figures, UMRs are shown as blue squares (placed at the middle of the identified segment), LMRs as red triangles (placed at the middle of the identified segment) and PMDs as green bars (extending over the entire PMD). The cut-off on methylation to determine UMRs and LMRs is shown as a grey dashed line.



Finally, the segmentation of UMRs and LMRs can be saved both as a GRanges object and a tab-delimited file as follows,

```
> saveUMRLMRSegments(segs=UMRLMRsegments.gr,
+ GRangesFilename="UMRsLMRs.gr.rds", TableFilename="UMRsLMRs.tab")
```

Here, `GRangesFilename` is the filename for the GRanges object and the `TableFilename` the filename for the tab-delimited file (only one of the two filenames is required).

References

[1] Michael B Stadler, Rabih Murr, Lukas Burger, Robert Ivanek, Florian Lienert, Anne Schöler, Erik van Nimwegen, Christiane Wirbelauer, Edward J Oakeley, Dimos Gaidatzis, Vijay K Tiwari, and Dirk Schübeler. DNA-binding factors shape the mouse methylome at distal regulatory regions. *Nature*, 480(7378):490–5, Dec 2011.

[2] Lukas Burger, Dimos Gaidatzis, Dirk Schubeler, and Michael B Stadler. Identification of active regulatory regions from DNA methylation data. *Nucleic Acids Res*, Jul 2013.

[3] Ryan Lister, Mattia Pelizzola, Robert H Dowen, R David Hawkins, Gary Hon, Julian Tonti-Filippini, Joseph R Nery, Leonard Lee, Zhen Ye, Que-Minh Ngo, Lee Edsall, Jessica Antosiewicz-Bourget, Ron Stewart, Victor Ruotti, A Harvey Millar, James A Thomson, Bing Ren, and Joseph R Ecker. Human DNA methylomes at base resolution show widespread epigenomic differences. *Nature*, 462(7271):315–22, Nov 2009.

[4] S T Sherry, M H Ward, M Kholodov, J Baker, L Phan, E M Smigielski, and K Sirotkin. dbSNP: the NCBI database of genetic variation. *Nucleic Acids Res*, 29(1):308–11, Jan 2001.