

# methyKit: User guide

Altuna Akalin

December 21, 2011

## 1 Introduction

In this manual, we will show how to use the methyKit package. methyKit is an R package for analysis and annotation of DNA methylation information obtained by high-throughput bisulfate sequencing. The package is designed to deal with sequencing data from RRBS and its variants. But it can potentially handle whole-genome bisulfate sequencing data if proper input format is provided.

## 2 Basics

### 2.1 Reading the methylation call files

We start by reading in the methylation call data from bisulfate sequencing with `read` function. Reading in the data this way will return a `methyRawList` object which stores methylation information per sample.

```
> library(methyKit)
> file.list=list( system.file("extdata", "test1.myCpG.txt", package = "methyKit"),
+               system.file("extdata", "test2.myCpG.txt", package = "methyKit"),
+               system.file("extdata", "control1.myCpG.txt", package = "methyKit"),
+               system.file("extdata", "control2.myCpG.txt", package = "methyKit") )
> # read the files to a methyRawList object: myobj
> myobj=read(file.list,
+           sample.id=list("test1","test2","ctrl1","ctrl2"),
+           assembly="hg18",
+           pipeline="amp",
+           treatment=c(1,1,0,0))
>
>
```

### 2.2 Descriptive statistics on samples

Since we read the data now, we can check the basic stats about the methylation data such as coverage and percent methylation. We now have a `methyRawList` object which contains methylation information per sample. The following command prints out percent methylation statistics for second sample: "test2"

```
> getMethylationStats(myobj[[2]],plot=F,both.strands=F)
```

methylation statistics per base

summary:

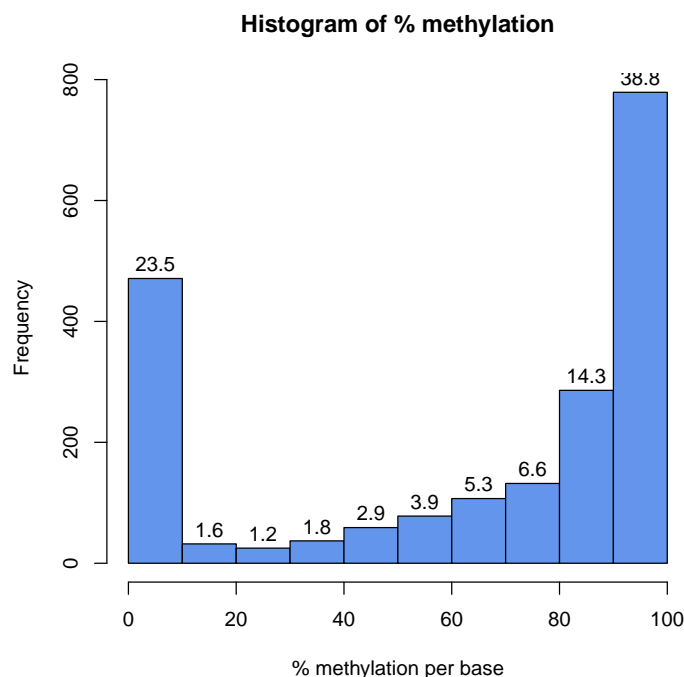
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.00	20.00	82.79	63.17	94.74	100.00

percentiles:

0%	10%	20%	30%	40%	50%	60%	70%
0.00000	0.00000	0.00000	48.38710	70.00000	82.78556	90.00000	93.33333
80%	90%	95%	99%	99.5%	99.9%	100%	
96.42857	100.00000	100.00000	100.00000	100.00000	100.00000	100.00000	

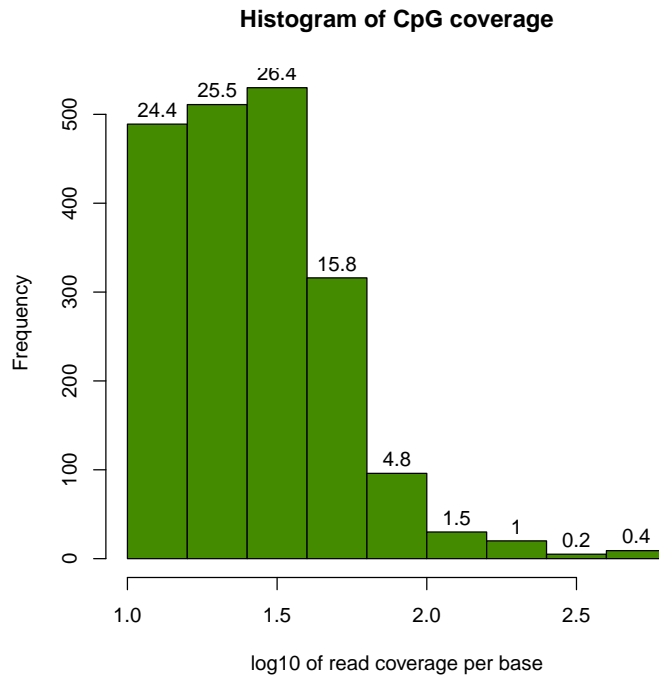
The following command plots the histogram for percent methylation distribution. The figure below is the histogram and numbers on bars denote what percentage of locations are contained in that bin. Typically, percent methylation histogram should have two peaks on both ends. In any given cell, any given base are either methylated or not. Therefore, looking at many cells should yield a similar pattern where we see lots of locations with high methylation and lots of locations with low methylation.

```
> library("graphics")
> getMethylationStats(myobj[[2]],plot=T,both.strands=F)
```



We can also plot the read coverage per base information in a similar way, again numbers on bars denote what percentage of locations are contained in that bin. Experiments that are highly suffering from PCR duplication bias will have a secondary peak towards the right hand side of the histogram.

```
> library("graphics")
> getCoverageStats(myobj[[2]],plot=T,both.strands=F)
```



### 2.3 Filtering samples based on read coverage

It might be useful to filter samples based on coverage. Particularly, if our samples are suffering from PCR bias it would be useful to discard bases with very high read coverage. Furthermore, we would also like to discard bases that have low read coverage, a high enough read coverage will increase the power of the statistical tests. The code below filters a `methylRawList` and discards bases that have coverage below 10X and also discards the bases that have more than 99.9th percentile of coverage in each sample.

```
> filtered.myobj=filterByCoverage(myobj,lo.count=10,lo.perc=NULL,
+                               hi.count=NULL,hi.perc=99.9)
```

## 3 Comparative analysis

### 3.1 Merging samples

In order to do further analysis, we will need to get the bases covered in all samples. The following function will merge all samples to one object for base-pair locations that are covered in all samples. Setting `destrand=TRUE` (the default is `FALSE`) will merge reads on both strands of a CpG dinucleotide. This provides better coverage, but only advised when looking at CpG methylation (for CpH methylation this will cause wrong results in subsequent analyses). This operation will return a `methylBase` object which will be our main object for all comparative analysis.

```
> methidh=unite(myobj,destrand=FALSE)
```

Let us take a look at the data content of methylBase object:

```
> head(methidh)
```

	id	chr	start	end	strand	coverage1	numCs1	numTs1
1	chr21.10011833	chr21	10011833	10011833	+	174	173	1
2	chr21.10011841	chr21	10011841	10011841	+	173	164	9
3	chr21.10011855	chr21	10011855	10011855	+	175	175	0
4	chr21.10011858	chr21	10011858	10011858	+	175	131	44
5	chr21.10011861	chr21	10011861	10011861	+	174	147	27
6	chr21.10011872	chr21	10011872	10011872	+	167	160	7

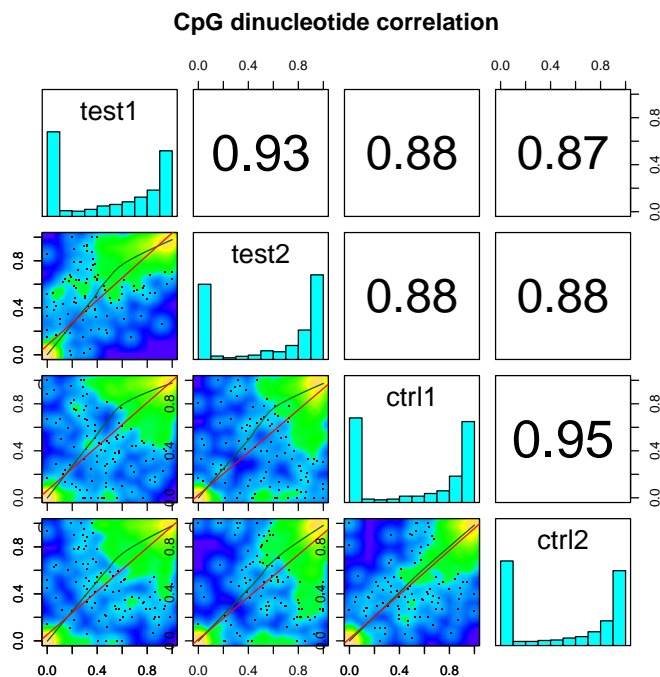
	coverage2	numCs2	numTs2	coverage3	numCs3	numTs3	coverage4	numCs4	numTs4
1	18	18	0	40	34	6	14	14	0
2	20	19	1	40	18	22	14	8	6
3	21	21	0	39	29	10	14	12	2
4	21	20	1	39	31	8	13	8	5
5	20	15	5	39	13	26	13	9	4
6	20	19	1	39	34	5	14	8	6

### 3.2 Sample Correlation

We can check the correlation between samples using `getCorrelation`. This function will either plot scatter plot and correlation coefficients or just print a correlation matrix

```
> getCorrelation(methidh,plot=T)
```

	test1	test2	ctrl1	ctrl2
test1	1.0000000	0.9252530	0.8767865	0.8737509
test2	0.9252530	1.0000000	0.8791864	0.8801669
ctrl1	0.8767865	0.8791864	1.0000000	0.9465369
ctrl2	0.8737509	0.8801669	0.9465369	1.0000000



### 3.3 Clustering samples

We can cluster the samples based on the similarity of their methylation profiles. The following function will cluster the samples and draw a dendrogram.

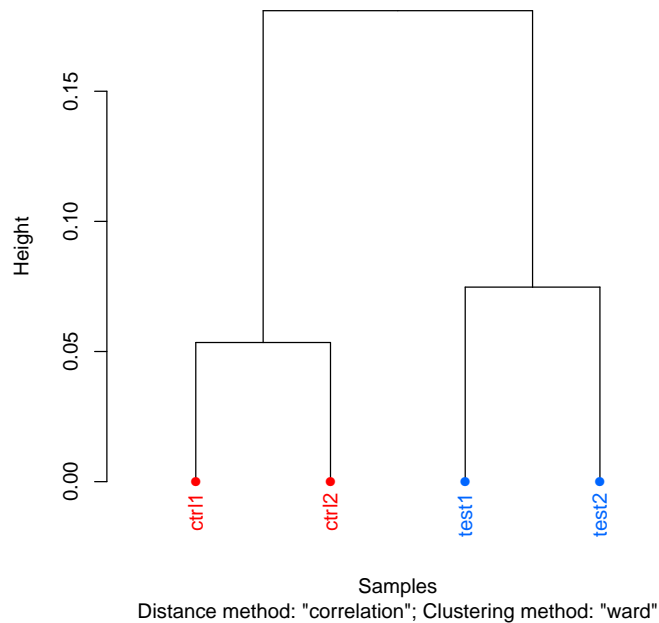
```
> clusterSamples(methidh, dist="correlation", method="ward", plot=TRUE)
```

Call:

```
hclust(d = d, method = HCLUST.METHODS[hclust.method])
```

```
Cluster method   : ward
Distance         : pearson
Number of objects: 4
```

### CpG dinucleotide methylation clustering



Setting the `plot=FALSE` will return a dendrogram object which can be manipulated by users or fed in to other user functions that can work with dendrograms.

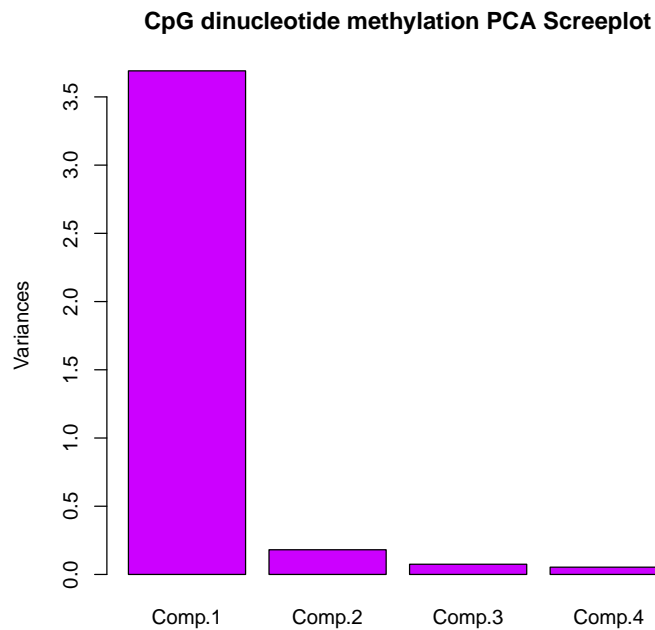
```
> hc = clusterSamples(methidh, dist="correlation", method="ward", plot=FALSE)
```

We can also do a PCA analysis on our samples. The following function will plot a scree plot for importance of components.

```
> PCASamples(methidh, screeplot=TRUE)
```

Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4
Standard deviation	1.9211651	0.42545636	0.2735654	0.23081050
Proportion of Variance	0.9227188	0.04525328	0.0187095	0.01331837
Cumulative Proportion	0.9227188	0.96797213	0.9866816	1.00000000

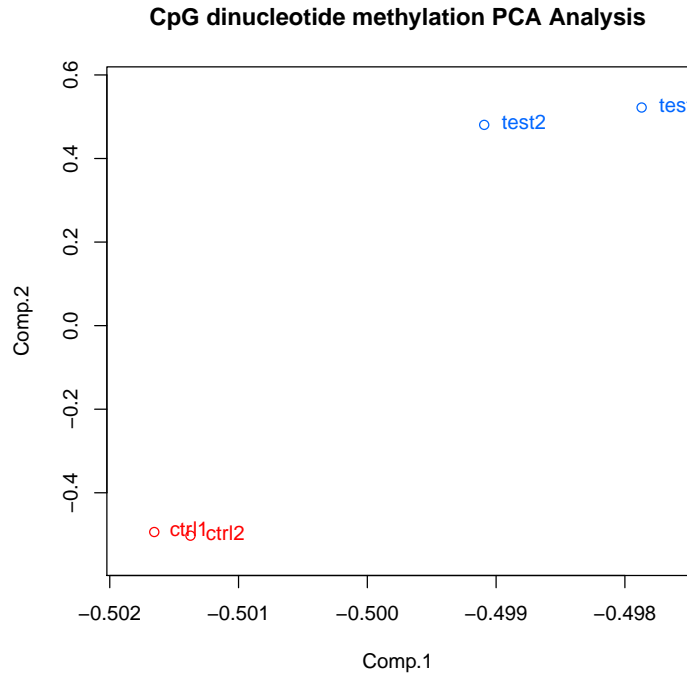


We can also plot PC1 and PC2 axis and a scatter plot of our samples on those axis which will reveal how they cluster.

```
> PCASamples(methidh)
```

Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4
Standard deviation	1.9211651	0.42545636	0.2735654	0.23081050
Proportion of Variance	0.9227188	0.04525328	0.0187095	0.01331837
Cumulative Proportion	0.9227188	0.96797213	0.9866816	1.00000000



### 3.4 Tiling windows analysis

For some situations, it might be desirable to summarize methylation information over tiling windows rather than doing base-pair resolution analysis. `methyKit` provides functionality to do such analysis. The function below tiles the genome with windows 1000bp length and 1000bp step-size and summarizes the methylation information on those tiles. In this case, it returns a `methyRawList` object which can be fed into `unite` and `calculateDiffMeth` functions to get differentially methylated regions

```
> tiles=tileMethylCounts(myobj,win.size=1000,step.size=1000)
> head(tiles[[1]])
```

	id	chr	start	end	strand	coverage	numCs	numTs
1	chr21.9764001.9765000	chr21	9764001	9765000	*	24	3	21
2	chr21.9820001.9821000	chr21	9820001	9821000	*	13	0	13
3	chr21.9837001.9838000	chr21	9837001	9838000	*	11	0	11
4	chr21.9849001.9850000	chr21	9849001	9850000	*	124	90	34
5	chr21.9853001.9854000	chr21	9853001	9854000	*	34	22	12
6	chr21.9860001.9861000	chr21	9860001	9861000	*	39	38	1

### 3.5 Finding differentially methylated bases or regions

`calculateDiffMeth()` function is the main function to calculate differential methylation. Depending on the sample size per each set it will either use Fisher's exact or logistic regression to calculate P-values. P-values will be adjusted to Q-values using SLIM method<sup>1</sup>.



```
> myDiff=calculateDiffMeth(methidh)
```

After q-value calculation, we can select the differentially methylated regions/bases based on q-value and percent methylation difference cutoffs. Following bit selects the bases that have q-value<0.01 and percent methylation difference larger than 25%.

```
> myDiff25p=get.methylDiff(myDiff,difference=25,qvalue=0.01)
```

We can also visualize the distribution of hypo/hyper-methylated bases/regions per chromosome using the following function. In this case, the example set includes only one chromosome. The list shows percentages of hypo/hyper methylated bases over all the covered bases in a given chromosome.

```
> diffMethPerChr(myDiff,plot=FALSE,qvalue.cutoff=0.01, meth.cutoff=25)
```

```
$diffMeth.per.chr
  chr number.of.hypomethylated percentage.of.hypomethylated
1 chr21                      59                      6.126687
  number.of.hpermethylated percentage.of.hpermethylated
1                        75                      7.788162

$diffMeth.all
  percentage.of.hpermethylated number.of.hpermethylated
1                        7.788162                      75
  percentage.of.hypomethylated number.of.hypomethylated
1                        6.126687                      59
```

## 4 Annotating differentially methylated bases or regions

We can annotate our differentially methylated regions/bases based on gene annotation. In this example, we read the gene annotation from a bed file and annotate our differentially methylated regions with that information. This will tell us what percentage of our differentially methylated regions are on promoters/introns/exons/intergenic region. Similar gene annotation can be fetched using `GenomicFeatures` package available from Bioconductor.org.

```
> gene.obj=read.transcript.features(system.file("extdata", "refseq.hg18.bed.txt",
+                                             package = "methylKit"))
> #
> # annotate differentially methylated Cs with promoter/exon/intron using annotation data
> #
> annotate.WithGenicParts(myDiff25p,gene.obj)
```

```
summary of target set annotation with genic parts
133 rows in target set
```

```
-----
-----
```

```
percentage of target features overlapping with annotation :
  promoter      exon      intron intergenic
```

```
27.81955    15.03759    34.58647    57.14286
```

```
percentage of target features overlapping with annotation (with promoter>exon>intron prece
promoter      exon      intron intergenic
27.81955      0.00000    15.03759    57.14286
```

```
percentage of annotation boundaries with feature overlap :
promoter      exon      intron
0.018129079 0.001589593 0.010038738
```

```
summary of distances to the nearest TSS :
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
5      828    45160    52030    94640   313500
```

Similarly, we can read the CpG island annotation and annotate our differentially methylated bases/regions with them.

```
> # read the shores and flanking regions and name the flanks as shores
> # and CpG islands as CpGi
> cpg.obj=read.feature.flank(system.file("extdata", "cpgi.hg18.bed.txt",
+                                     package = "methylKit"),
+                             feature.flank.name=c("CpGi","shores"))
> #
> #
> diffCpGann=annotate.WithFeature.Flank(myDiff25p,cpg.obj$CpGi,cpg.obj$shores,
+                                       feature.name="CpGi",flank.name="shores")
```

## 4.1 Regional analysis

We can also summarize methylation information over a set of defined regions such as promoters or CpG islands. The function below summarizes the methylation information over a given set of promoter regions and outputs a `methylRaw` or `methylRawList` object depending on the input.

```
> promoters=regionCounts(myobj,gene.obj$promoters)
> head(promoters[[1]])
```

	id	chr	start	end	strand	coverage	numCs
1	chr21.17806094.17808094.NA	chr21	17806094	17808094	+	1834	7
2	chr21.10119796.10121796.NA	chr21	10119796	10121796	-	79	44
3	chr21.10011791.10013791.NA	chr21	10011791	10013791	-	3697	2982
4	chr21.10119808.10121808.NA	chr21	10119808	10121808	-	79	44
5	chr21.15357997.15359997.NA	chr21	15357997	15359997	-	8613	16
6	chr21.16023366.16025366.NA	chr21	16023366	16025366	+	6296	5

```
numTs
1 1827
2 35
3 715
```

```

4    35
5   8594
6   6291

```

## 4.2 Convenience functions for annotation objects

After getting the annotation of differentially methylated regions, we can get the distance to TSS and nearest gene name using the `getAssociationWithTSS` function.

```

> diffAnn=annotate.WithGenicParts(myDiff25p, gene.obj)
> # target.row is the row number in myDiff25p
> head(getAssociationWithTSS(diffAnn))

```

	target.row	dist.to.feature	feature.name	feature.strand
60	1	951	NM_199260	-
60.1	2	931	NM_199260	-
60.2	3	838	NM_199260	-
60.3	4	828	NM_199260	-
60.4	5	802	NM_199260	-
60.5	6	723	NM_199260	-

It is also desirable to get percentage/number of differentially methylated regions that overlap with intron/exon/promoters

```

> getTargetAnnotationStats(diffAnn, percentage=TRUE, precedence=TRUE)

```

promoter	exon	intron	intergenic
27.81955	0.00000	15.03759	57.14286

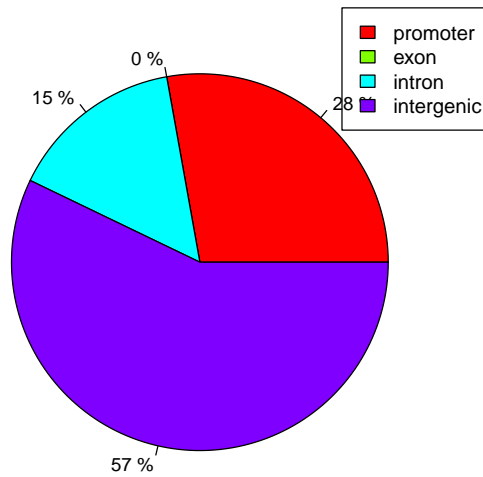
We can also plot the percentage of differentially methylated bases overlapping with exon/intron/promoters

```

> plotTargetAnnotation(diffAnn, precedence=TRUE,
+   main="differential methylation annotation")

```

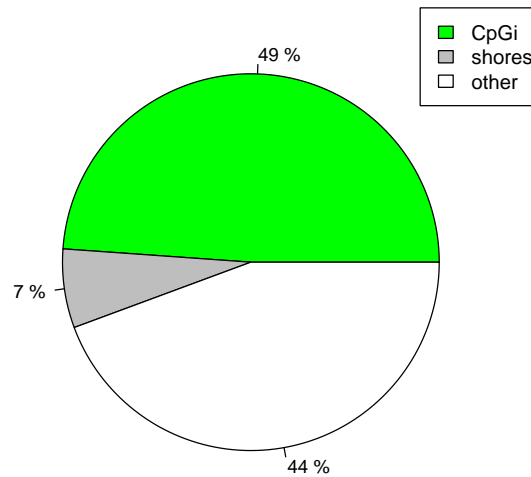
### differential methylation annotation



We can also plot the CpG island annotation the same way.

```
> plotTargetAnnotation(diffCpGann,col=c("green","gray","white"),  
+   main="differential methylation annotation")
```

### differential methylation annotation



It might be also useful to get percentage of intron/exon/promoters that overlap with differentially methylated bases.

```
> getFeatsWithTargetsStats(diffAnn,percentage=FALSE)
```

promoter	exon	intron
5	6	34

## References

- [1] Hong-Qiang Wang, Lindsey K Tuominen, and Chung-Jui Tsai. SLIM: a sliding linear model for estimating the proportion of true null hypotheses in datasets with dependence structures. *Bioinformatics (Oxford, England)*, 27(2):225–31, January 2011.