

# Package ‘sesame’

March 30, 2021

**Type** Package

**Title** Tools For Analyzing Illumina Infinium DNA Methylation Arrays

**Description**

Tools For analyzing Illumina Infinium DNA methylation arrays. SeSAmE provides utilities to support analyses of multiple generations of Infinium DNA methylation BeadChips, including pre-processing, quality control, visualization and inference. SeSAmE features more accurate detection calling, intelligent inference of ethnicity, sex and advanced quality control routines.

**Version** 1.8.2

**Depends** R (>= 4.0), sesameData, methods

**License** MIT + file LICENSE

**RoxygenNote** 7.1.1

**Imports** BiocParallel, grDevices, utils, illuminaio, MASS,  
GenomicRanges, IRanges, grid, preprocessCore, stats, S4Vectors,  
randomForest, wheatmap, ggplot2, parallel, matrixStats,  
DNAcopy, HDF5Array, SummarizedExperiment

**Suggests** scales, knitr, rmarkdown, testthat, minfi,  
FlowSorted.CordBloodNorway.450k, FlowSorted.Blood.450k, dplyr,  
tidyr, BiocStyle

**Encoding** UTF-8

**VignetteBuilder** knitr

**URL** <https://github.com/zwdzwd/sesame>

**BugReports** <https://github.com/zwdzwd/sesame/issues>

**biocViews** DNAMethylation, MethylationArray, Preprocessing,  
QualityControl

**Collate** 'GEO.R' 'QC.R' 'SigSetList.R' 'sesame.R' 'SigSetMethods.R'  
'age.R' 'background\_correction.R' 'cell\_composition.R'  
'channel\_inference.R' 'cnv.R' 'controls.R' 'deidentify.R'  
'detection.R' 'differential\_methylation.R' 'dye\_bias.R'  
'fileSet.R' 'open.R' 'sesamize.R' 'simulate\_data.R' 'snp.R'  
'track.R' 'utils.R' 'vcf.R' 'visualize.R'

**git\_url** <https://git.bioconductor.org/packages/sesame>

**git\_branch** RELEASE\_3\_12

**git\_last\_commit** af4eb84

**git\_last\_commit\_date** 2020-11-13

**Date/Publication** 2021-03-29

**Author** Wanding Zhou [aut, cre],  
 Hui Shen [aut],  
 Timothy Triche [ctb],  
 Bret Barnes [ctb]

**Maintainer** Wanding Zhou <zhouwanding@gmail.com>

## R topics documented:

sesame-package . . . . .	4
as.data.frame.sesameQC . . . . .	5
BetaValueToMValue . . . . .	5
binSignals . . . . .	6
bisConversionControl . . . . .	6
buildControlMatrix450k . . . . .	7
chipAddressToSignal . . . . .	7
cnSegmentation . . . . .	8
createUCSCtrack . . . . .	9
ctl . . . . .	9
ctl<- . . . . .	10
deIdentify . . . . .	11
detectionMask . . . . .	11
detectionPfixedNorm . . . . .	12
detectionPnegEcdf . . . . .	13
detectionPnegNorm . . . . .	13
detectionPnegNormGS . . . . .	14
detectionPnegNormTotal . . . . .	15
detectionPoobEcdf . . . . .	15
detectionZero . . . . .	16
diffRefSet . . . . .	17
DML . . . . .	17
DMR . . . . .	18
dyeBiasCorr . . . . .	19
dyeBiasCorrMostBalanced . . . . .	20
dyeBiasCorrTypeINorm . . . . .	20
estimateCellComposition . . . . .	21
estimateLeukocyte . . . . .	21
extra . . . . .	22
extra<- . . . . .	23
formatVCF . . . . .	23
getAFTypeIbySumAlleles . . . . .	24
getBetas . . . . .	25
getBinCoordinates . . . . .	25
getNormCtls . . . . .	26
getProbesByGene . . . . .	26
getProbesByRegion . . . . .	27
getProbesByTSS . . . . .	28
getRefSet . . . . .	28
getSegment . . . . .	29
getSexInfo . . . . .	30
IG . . . . .	30

IG<-	31
II	31
II<-	32
inferEthnicity	33
inferSex	33
inferSexKaryotypes	34
inferTypeIChannel	35
initFileSet	35
IR	36
IR<-	37
makeExampleSeSAMEDataSet	37
makeExampleTinyEPICDataSet	38
mapFileSet	38
meanIntensity	39
MValueToBetaValue	40
noob	40
noobsb	41
oobG	41
oobG<-	42
oobR	42
oobR<-	43
openSesame	44
openSesameToFile	45
parseGEOSignalABFile	45
predictAgeHorvath353	46
predictAgePheno	46
predictAgeSkinBlood	47
print.fileSet	48
print.sesameQC	48
probeNames	49
pval	49
pval<-	50
qualityMask	51
qualityRank	51
readFileSet	52
readIDATpair	53
reIdentify	53
reopenSesame	54
RGChannelSetToSigSets	55
searchIDATprefixes	55
segmentBins	56
sesameQC	56
sesamize	57
show,SigSet-method	58
signalMU	58
SigSet-class	59
SigSetList	60
SigSetList-class	60
SigSetList-methods	60
SigSetListFromIDATs	61
SigSetListFromPath	61
SigSetsToRGChannelSet	62

SigSetToRatioSet . . . . .	63
sliceFileSet . . . . .	63
SNPcheck . . . . .	64
subsetSignal . . . . .	64
topLoci . . . . .	65
topSegments . . . . .	66
totalIntensities . . . . .	66
totalIntensityZscore . . . . .	67
twoCompsEst2 . . . . .	67
visualizeGene . . . . .	68
visualizeProbes . . . . .	69
visualizeRegion . . . . .	70
visualizeSegments . . . . .	71

<b>Index</b>	<b>72</b>
--------------	-----------

---

sesame-package	<i>Analyze DNA methylation data</i>
----------------	-------------------------------------

---

## Description

Sensible and step-wise analysis of DNA methylation data

## Details

This package complements array functionalities that allow processing >10,000 samples in parallel on clusters.

## Author(s)

Wanding Zhou <Wanding.Zhou@vai.org>, Hui Shen <Hui.Shen@vai.org> Timothy J Triche Jr <Tim.Triche@vai.org>

## See Also

Useful links:

- <https://github.com/zwdzwd/sesame>
- Report bugs at <https://github.com/zwdzwd/sesame/issues>

## Examples

```
sset <- readIDATpair(sub('_Grn.idat','',system.file(
  'extdata','4207113116_A_Grn.idat',package='sesameData'))))

## The OpenSesame pipeline
betas <- openSesame(sset)
```

---

`as.data.frame.sesameQC`*Coerce a sesameQC into a dataframe*

---

**Description**

Coerce a sesameQC into a dataframe

**Usage**

```
## S3 method for class 'sesameQC'  
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

**Arguments**

<code>x</code>	a sesameQC object
<code>row.names</code>	see <code>as.data.frame</code>
<code>optional</code>	see <code>as.data.frame</code>
<code>...</code>	see <code>as.data.frame</code>

**Value**

a data.frame

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset  
qc <- sesameQC(sset)  
df <- as.data.frame(qc)
```

---

`BetaValueToMValue`      *Convert beta-value to M-value*

---

**Description**

Logit transform a beta value vector to M-value vector.

**Usage**

```
BetaValueToMValue(b)
```

**Arguments**

<code>b</code>	vector of beta values
----------------	-----------------------

**Details**

Convert beta-value to M-value (aka logit transform)

**Value**

a vector of M values

**Examples**

```
BetaValueToMValue(c(0.1, 0.5, 0.9))
```

---

binSignals	<i>Bin signals from probe signals</i>
------------	---------------------------------------

---

**Description**

require GenomicRanges

**Usage**

```
binSignals(probe.signals, bin.coords, probe.coords)
```

**Arguments**

probe.signals	probe signals
bin.coords	bin coordinates
probe.coords	probe coordinates

**Value**

bin signals

---

bisConversionControl	<i>Compute internal bisulfite conversion control</i>
----------------------	--

---

**Description**

Compute GCT score for internal bisulfite conversion control. The function takes a SigSet as input. The higher the GCT score, the more likely the incomplete conversion.

**Usage**

```
bisConversionControl(sset, use.median = FALSE)
```

**Arguments**

sset	signal set
use.median	use median to compute GCT instead of mean

**Value**

GCT score (the higher, the more incomplete conversion)

**Examples**

```
sset <- makeExampleSeSAMEDataSet('HM450')
bisConversionControl(sset)
```

---

```
buildControlMatrix450k
```

*Build control summary matrix*

---

**Description**

The function takes a SigSet as input and outputs the control matrix summary vector. This vector summarizes one single QC metric for the array control. This includes bisulfite control, stain signal extension efficiency and more.

**Usage**

```
buildControlMatrix450k(sset)
```

**Arguments**

sset                    an object of class SigSet

**Value**

a vector with control summaries

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
control.summary <- buildControlMatrix450k(sset)
```

---

```
chipAddressToSignal    Lookup address in one sample
```

---

**Description**

Lookup address and transform address to probe

**Usage**

```
chipAddressToSignal(dm, manifest, controls = NULL)
```

**Arguments**

dm                    data frame in chip address, 2 columns: cy3/Grn and cy5/Red  
manifest              a data frame with columns Probe\_ID, M, U and col  
controls              a data frame with columns Address and Name. This is optional but might be necessary for some preprocessing methods that depends on these control probes. This is left for backward compatibility. Updated version should have controls consolidated into manifest.

**Details**

Translate data in chip address to probe address. Type I probes can be separated into Red and Grn channels. The methylated allele and unmethylated allele are at different addresses. For type II probes methylation allele and unmethylated allele are at the same address. Grn channel is for methylated allele and Red channel is for unmethylated allele. The out-of-band signals are type I probes measured using the other channel.

**Value**

a SigSet, indexed by probe ID address

---

cnSegmentation	<i>Perform copy number segmentation</i>
----------------	---

---

**Description**

Perform copy number segmentation using the signals in the signal set. The function takes a SigSet for the target sample and a set of normal SigSet for the normal samples. An optional arguments specifies the version of genome build that the inference will operate on. The function outputs an object of class CNSegment with signals for the segments ( seg.signals), the bin coordinates ( bin.coords) and bin signals (bin.signals).

**Usage**

```
cnSegmentation(sset, ssets.normal, refversion = c("hg19", "hg38"))
```

**Arguments**

sset	SigSet
ssets.normal	SigSet for normalization
refversion	hg19 or hg38

**Value**

an object of CNSegment

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
ssets.normal <- sesameDataGet('EPIC.5.normal')
seg <- cnSegmentation(sset, ssets.normal)
```



---

createUCSCtrack	<i>Turn beta values into a UCSC browser track</i>
-----------------	---

---

**Description**

Turn beta values into a UCSC browser track

**Usage**

```
createUCSCtrack(betas, output = NULL, platform = "HM450", refversion = "hg38")
```

**Arguments**

betas	a named numeric vector
output	output file name
platform	HM450, EPIC etc.
refversion	hg38, hg19 etc.

**Value**

when output is null, return a data.frame, otherwise NULL

**Examples**

```
betas.tissue <- sesameDataGet('HM450.1.TCGA.PAAD')$betas
## add output to create an actual file
df <- createUCSCtrack(betas.tissue)

## to convert to bigBed
## sort -k1,1 -k2,2n output.bed >output_sorted.bed
## bedToBigBed output_sorted.bed hg38.chrom output.bb
```

---

ctl	<i>ctl getter generic</i>
-----	---------------------------

---

**Description**

ctl getter generic  
Get ctl slot of SigSet class

**Usage**

```
ctl(x)

## S4 method for signature 'SigSet'
ctl(x)
```

**Arguments**

x                    object of SigSet

**Value**

The ctl slot of SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
head(ctl(sset))
```

---

ctl<-                    *ctl replacement generic*

---

**Description**

ctl replacement generic

Replace ctl slot of SigSet class

**Usage**

```
ctl(x) <- value
```

```
## S4 replacement method for signature 'SigSet'
ctl(x) <- value
```

**Arguments**

x                    object of SigSet

value                new value

**Value**

a new SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
df <- ctl(sset)
df[1,1] <- 10
ctl(sset) <- df
```

---

deIdentify	<i>De-identify IDATs by removing SNP probes</i>
------------	---

---

**Description**

Mask SNP probe intensity mean by zero.

**Usage**

```
deIdentify(path, out_path = NULL, snps = NULL, mft = NULL, randomize = FALSE)
```

**Arguments**

path	input IDAT file
out_path	output IDAT file
snps	SNP definition, if not given, default to SNP probes
mft	sesame-compatible manifest if non-standard
randomize	whether to randomize the SNPs. if TRUE, randomize the signal intensities. one can use set.seed to reidentify the IDAT with the secret seed (see examples). If FALSE, this sets all SNP intensities to zero.

**Value**

NULL, changes made to the IDAT files

**Examples**

```
my_secret <- 13412084
set.seed(my_secret)
temp_out <- tempfile("test")
deIdentify(system.file(
  "extdata", "4207113116_A_Grn.idat", package = "sesameData"),
  temp_out, randomize = TRUE)
unlink(temp_out)
```

---

detectionMask	<i>Mask Sigset by detection p-value</i>
---------------	---

---

**Description**

Mask Sigset by detection p-value

**Usage**

```
detectionMask(sset, pval.method = NULL, pval.threshold = 0.05)
```

**Arguments**

sset                    a SigSet  
 pval.method        which method to use in calculating p-values  
 pval.threshold    the p-value threshold

**Value**

a filtered SigSet

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
sset.masked <- detectionMask(sset)
```

---

detectionPfixedNorm    *Detection P-value based on normal fitting with gived parameters*

---

**Description**

The function takes a SigSet as input, computes detection p-value using negative control probes parametrized in a normal distribution and returns a new SigSet with an updated pval slot.

**Usage**

```
detectionPfixedNorm(  

  sset,  

  muG = 500,  

  sdG = 200,  

  muR = 500,  

  sdR = 200,  

  force = FALSE  

)
```

**Arguments**

sset                    a SigSet  
 muG                    mean of background in Grn channel  
 sdG                    SD of background in Grn channel  
 muR                    mean of background in Red channel  
 sdR                    SD of background in Red channel  
 force                  force rerun even if result already exists

**Details**

Background of Grn and Red are estimated separately from a fixed normal distribution. p-value is taken from the minimum of the p-value of the two alleles (color depends on probe design).

**Value**

detection p-value

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
sset <- detectionPfixedNorm(sset)
```

---

detectionPnegEcdf      *Detection P-value based on ECDF of negative control*

---

**Description**

The function takes a SigSet as input, computes detection p-value using negative control probes' empirical distribution and returns a new SigSet with an updated pval slot.

**Usage**

```
detectionPnegEcdf(sset, force = FALSE)
```

**Arguments**

sset	a SigSet
force	force rerun even if result already exists

**Value**

detection p-value

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
sset <- detectionPnegEcdf(sset)
```

---

detectionPnegNorm      *Detection P-value based on normal fitting the negative controls*

---

**Description**

The function takes a SigSet as input, computes detection p-value using negative control probes parametrized in a normal distribution and returns a new SigSet with an updated pval slot.

**Usage**

```
detectionPnegNorm(sset, force = FALSE)
```

**Arguments**

sset	a SigSet
force	force rerun even if result already exists

**Details**

Background of Grn and Red are estimated separately from negative control probes-parameterized normal distribution. p-value is taken from the minimum of the p-value of the two alleles (color depends on probe design).

**Value**

detection p-value

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
sset <- detectionPnegNorm(sset)
```

---

detectionPnegNormGS     *Detection P-value emulating Genome Studio*

---

**Description**

The function takes a SigSet as input, computes detection p-value using negative control probes parametrized in a normal distribution a la Genome Studio and returns a new SigSet with an updated pval slot.

**Usage**

```
detectionPnegNormGS(sset, force = FALSE)
```

**Arguments**

sset	a SigSet
force	force rerun even if result already exists

**Details**

P-value is calculated using negative control probes as the estimate of background where Grn channel and Red channel are merged. But when estimating p-value the Red and Grn are summed (non-ideal).

**Value**

detection p-value

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
sset <- detectionPnegNormGS(sset)
```

---

detectionPnegNormTotal

*Detection P-value based on normal fitting the negative controls, channels are first summed*

---

### Description

The function takes a SigSet as input, computes detection p-value using negative control probes parametrized in a normal distribution with the two channels summed first and returns a new SigSet with an updated pval slot. The SD is summed to emulate the SD of the summed signal (not the most accurate treatment).

### Usage

```
detectionPnegNormTotal(sset, force = FALSE)
```

### Arguments

sset	a SigSet
force	force rerun even if result already exists

### Value

detection p-value

### Examples

```
sset <- makeExampleSeSAMEDataSet()
sset <- detectionPnegNormTotal(sset)
```

---

detectionPoobEcdf

*Detection P-value based on ECDF of out-of-band signal*

---

### Description

aka pOOBAH (p-vals by Out-Of-Band Array Hybridization)

### Usage

```
detectionPoobEcdf(sset, force = FALSE)
```

```
pOOBAH(sset, force = FALSE)
```

### Arguments

sset	a SigSet
force	force rerun even if result already exists

**Details**

The function takes a SigSet as input, computes detection p-value using out-of-band probes empirical distribution and returns a new SigSet with an updated pval slot.

**Value**

detection p-value

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
sset <- detectionPoobEcdf(sset)

sset <- makeExampleSeSAMEDataSet()
sset <- pOOBAH(sset)
```

---

detectionZero

*Detection P-value set to all zero*

---

**Description**

Detection P-value set to all zero

**Usage**

```
detectionZero(sset, force = FALSE)
```

**Arguments**

sset	a SigSet
force	force rerun even if result already exists

**Value**

detection p-value set to all zero

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
sset <- detectionZero(sset)
```



---

diffRefSet	<i>Restrict refset to differentially methylated probes use with care, might introduce bias</i>
------------	--

---

### Description

The function takes a matrix with probes on the rows and cell types on the columns and output a subset matrix and only probes that show discordant methylation levels among the cell types.

### Usage

```
diffRefSet(g)
```

### Arguments

`g` a matrix with probes on the rows and cell types on the columns

### Value

`g` a matrix with a subset of input probes (rows)

### Examples

```
g <- diffRefSet(getRefSet(platform='HM450'))
```

---

DML	<i>Test differential methylation on each locus</i>
-----	--

---

### Description

The function takes a beta value matrix with probes on the rows and samples on the columns. It also takes a sample information data frame (`sample.data`) and formula for testing. The function outputs a list of coefficient tables for each factor tested.

### Usage

```
DML(  
  betas,  
  sample.data,  
  formula,  
  se.lb = 0.06,  
  balanced = FALSE,  
  cf.test = NULL  
)
```

**Arguments**

betas	beta values
sample.data	data frame for sample information, column names are predictor variables (e.g., sex, age, treatment, tumor/normal etc) and are referenced in formula. Rows are samples.
formula	formula
se.lb	lower bound to standard error of slope, lower this to get more difference of small effect size.
balanced	whether design is balanced or not. default to FALSE, when unbalanced will use Welch's method to estimate standard error. balance=TRUE is faster.
cf.test	factors to test (default to all factors in formula except intercept). Use "all" for all factors.

**Value**

cf - a list of coefficient tables for each factor

**Examples**

```
data <- sesameDataGet('HM450.76.TCGA.matched')
cf <- DML(data$betas, data$sampleInfo, ~type)
```

---

DMR

---

*Find Differentially Methylated Region (DMR)*


---

**Description**

This subroutine uses Euclidean distance to group CpGs and then combine p-values for each segment. The function performs DML test first if cf is NULL. It groups the probe testing results into differential methylated regions in a coefficient table with additional columns designating the segment ID and statistical significance (P-value) testing the segment.

**Usage**

```
DMR(
  betas,
  sample.data = NULL,
  formula = NULL,
  cf = NULL,
  dist.cutoff = NULL,
  seg.per.locus = 0.5,
  platform = c("EPIC", "HM450"),
  refversion = c("hg38", "hg19"),
  ...
)
```

**Arguments**

betas	beta values for distance calculation
sample.data	data frame for sample information, column names are predictor variables (e.g., sex, age, treatment, tumor/normal etc) and are referenced in formula. Rows are samples.
formula	formula
cf	coefficient table from diffMeth, when NULL will be computed from beta. If cf is given, sample.data and formula are ignored.
dist.cutoff	distance cutoff (default to use dist.cutoff.quantile)
seg.per.locus	number of segments per locus higher value leads to more segments
platform	EPIC or HM450
refversion	hg38 or hg19
...	additional parameters to DML

**Value**

coefficient table with segment ID and segment P-value

**Examples**

```
data <- sesameDataGet('HM450.76.TCGA.matched')
cf <- DMR(data$betas, data$sampleInfo, ~type)
```

---

dyeBiasCorr                      *Correct dye bias in by linear scaling.*

---

**Description**

The function takes a SigSet as input and scale both the Grn and Red signal to a reference (ref) level. If the reference level is not given, it is set to the mean intensity of all the in-band signals. The function returns a SigSet with dye bias corrected.

**Usage**

```
dyeBiasCorr(sset, ref = NULL)
```

**Arguments**

sset	a SigSet
ref	reference signal level

**Value**

a normalized SigSet

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
sset.db <- dyeBiasCorr(sset)
```

---

dyeBiasCorrMostBalanced

*Correct dye bias using most balanced sample as the reference*

---

### Description

The function chose the reference signal level from a list of SigSet. The chosen sample has the smallest difference in Grn and Red signal intensity as measured using the normalization control probes. In practice, it doesn't matter which sample is chosen as long as the reference level does not deviate much. The function returns a list of SigSets with dye bias corrected.

### Usage

```
dyeBiasCorrMostBalanced(ssets)
```

### Arguments

ssets            a list of normalized SigSets

### Value

a list of normalized SigSets

### Examples

```
ssets <- sesameDataGet('HM450.10.TCGA.BLCA.normal')
ssets.db <- dyeBiasCorrMostBalanced(ssets)
```

---

dyeBiasCorrTypeINorm    *Dye bias correction by matching green and red to mid point*


---

### Description

This function compares the Type-I Red probes and Type-I Grn probes and generates and mapping to correct signal of the two channels to the middle. The function takes one single SigSet and returns a SigSet with dye bias corrected.

### Usage

```
dyeBiasCorrTypeINorm(sset)
```

### Arguments

sset            a SigSet

### Value

a SigSet after dye bias correction.

### Examples

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
sset.db <- dyeBiasCorrTypeINorm(sset)
```

---

 estimateCellComposition

*Estimate cell composition using reference*


---

### Description

This is a reference-based cell composition estimation. The function takes a reference methylation status matrix (rows for probes and columns for cell types, can be obtained by getRefSet function) and a query beta value measurement. The length of the target beta values should be the same as the number of rows of the reference matrix. The method assumes one unknown component. It outputs a list containing the estimated cell fraction, the error of optimization and methylation status of the unknown component.

### Usage

```
estimateCellComposition(g, q, refine = TRUE, dichotomize = FALSE, ...)
```

### Arguments

g	reference methylation
q	target measurement: length(q) == nrow(g)
refine	to refine estimate, takes longer
dichotomize	to dichotomize query beta value before estimate, this relieves unclean background subtraction
...	extra parameters for optimization, this includes temp - annealing temperature (0.5) maxIter - maximum iteration to stop after converge (1000) delta - delta score to reset counter (0.0001) verbose - output debug info (FALSE)

### Value

a list of fraction, min error and unknown component methylation state

---

estimateLeukocyte

*Estimate leukocyte fraction using a two-component model*


---

### Description

The method assumes only two components in the mixture: the leukocyte component and the target tissue component. The function takes the beta values matrix of the target tissue and the beta value matrix of the leukocyte. Both matrices have probes on the row and samples on the column. Row names should have probe IDs from the platform. The function outputs a single numeric describing the fraction of leukocyte.

### Usage

```
estimateLeukocyte(
  betas.tissue,
  betas.leuko = NULL,
  betas.tumor = NULL,
  platform = c("EPIC", "HM450", "HM27")
)
```

**Arguments**

betas.tissue    tissue beta value matrix (#probes X #samples)  
 betas.leuko    leukocyte beta value matrix, if missing, use the SeSAmE default by infinium platform  
 betas.tumor    optional, tumor beta value matrix  
 platform       "HM450", "HM27" or "EPIC"

**Value**

leukocyte estimate, a numeric vector

**Examples**

```
betas.tissue <- sesameDataGet('HM450.1.TCGA.PAAD')$betas
estimateLeukocyte(betas.tissue)
```

---

 extra

*extra getter generic*


---

**Description**

extra getter generic  
 Get extra slot of SigSet class

**Usage**

```
extra(x)

## S4 method for signature 'SigSet'
extra(x)
```

**Arguments**

x                    object of SigSet

**Value**

The extra slot of SigSet

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
head(extra(sset))
```

---

extra<-                    *extra replacement generic*

---

**Description**

extra replacement generic  
 Replace extra slot of SigSet class

**Usage**

```
extra(x) <- value

## S4 replacement method for signature 'SigSet'
extra(x) <- value
```

**Arguments**

x	object of SigSet
value	new value

**Value**

a new SigSet

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
df <- extra(sset)
extra(sset) <- list(pval=numeric(0))
```

---

formatVCF                    *Convert SNP from Infinium array to VCF file*

---

**Description**

Convert SNP from Infinium array to VCF file

**Usage**

```
formatVCF(sset, vcf = NULL, refversion = "hg19", annoS = NULL, annoI = NULL)
```

**Arguments**

sset	SigSet
vcf	output VCF file path, if NULL output to console
refversion	reference version, currently only support
annoS	SNP variant annotation, download if not given
annoI	Infinium-I variant annotation, download if not given hg19 and hg38 in human

**Value**

VCF file. If vcf is NULL, a data.frame is output to console. The data.frame does not contain VCF headers.

Note the vcf is not sorted. You can sort with `awk '$1 ~ /^#/ print $0;next print $0 | "sort -k1,1 -k2,2n"'`

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset

annoS <- sesameDataPullVariantAnno_SNP('EPIC', 'hg19')
annoI <- sesameDataPullVariantAnno_InfiniumI('EPIC', 'hg19')
## output to console
head(formatVCF(sset, annoS=annoS, annoI=annoI))
```

---

getAFTypeIbySumAlleles

*Get allele frequency treating type I by summing alleles*

---

**Description**

Takes a SigSet as input and returns a numeric vector containing extra allele frequencies based on Color-Channel-Switching (CCS) probes. If no CCS probes exist in the SigSet, then an numeric(0) is returned.

**Usage**

```
getAFTypeIbySumAlleles(sset, known.ccs.only = TRUE)
```

**Arguments**

sset                    SigSet  
known.ccs.only    consider only known CCS probes

**Value**

beta values

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
betas <- getAFTypeIbySumAlleles(sset)
```



---

getBetas	<i>Get beta Values</i>
----------	------------------------

---

**Description**

sum.typeI is used for rescuing beta values on Color-Channel-Switching CCS probes. The function takes a SigSet and returns beta value except that Type-I in-band signal and out-of-band signal are combined. This prevents color-channel switching due to SNPs.

**Usage**

```
getBetas(sset, mask = TRUE, sum.TypeI = FALSE)
```

**Arguments**

sset	SigSet
mask	whether to use mask
sum.TypeI	whether to sum type I channels

**Value**

a numeric vector, beta values

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset  
betas <- getBetas(sset)
```

---

getBinCoordinates	<i>Get bin coordinates</i>
-------------------	----------------------------

---

**Description**

requires GenomicRanges, IRanges

**Usage**

```
getBinCoordinates(seqInfo, gapInfo, probe.coords)
```

**Arguments**

seqInfo	chromosome information object
gapInfo	chromosome gap information
probe.coords	probe coordinates

**Value**

bin.coords

---

getNormCtrls	<i>get normalization control signal</i>
--------------	---

---

**Description**

get normalization control signal from SigSet. The function optionally takes mean for each channel.

**Usage**

```
getNormCtrls(sset, average = FALSE)
```

**Arguments**

sset	a SigSet
average	whether to average

**Value**

a data frame of normalization control signals

**Examples**

```
sset <- readIDATpair(file.path(system.file(
  'extdata', '', package='sesameData'), '4207113116_B'))

df.ctrl <- getNormCtrls(sset)
```

---

getProbesByGene	<i>Get Probes by Gene</i>
-----------------	---------------------------

---

**Description**

Get probes mapped to a gene. All transcripts for the gene are considered. The function takes a gene name as appears in UCSC RefGene database. The platform and reference genome build can be changed with 'platform' and 'refversion' options. The function returns a vector of probes that falls into the given gene.

**Usage**

```
getProbesByGene(
  geneName,
  platform = c("EPIC", "HM450"),
  upstream = 0,
  downstream = 0,
  refversion = c("hg38", "hg19")
)
```

**Arguments**

geneName	gene name
platform	EPIC or HM450
upstream	number of bases to expand upstream of target gene
dwestream	number of bases to expand downstream of target gene
refversion	hg38 or hg19

**Value**

probes that fall into the given gene

**Examples**

```
probes <- getProbesByGene('CDKN2A', upstream=500, dwestream=500)
```

---

getProbesByRegion      *Get probes by genomic region*

---

**Description**

The function takes a genomic coordinate and output the a vector of probes on the specified platform that falls in the given genomic region.

**Usage**

```
getProbesByRegion(
  chrm,
  beg = 1,
  end = -1,
  platform = c("EPIC", "HM450"),
  refversion = c("hg38", "hg19")
)
```

**Arguments**

chrm	chromosome
beg	begin, 1 if omitted
end	end, chromosome end if omitted
platform	EPIC or HM450
refversion	hg38 or hg19

**Value**

probes that fall into the given region

**Examples**

```
getProbesByRegion('chr5', 135413937, 135419936,
  refversion = 'hg19', platform = 'HM450')
```

---

getProbesByTSS                      *Get Probes by Gene Transcription Start Site (TSS)*

---

### Description

Get probes mapped to a TSS. All transcripts for the gene are considered. The function takes a gene name as appears in UCSC RefGene database. The platform and reference genome build can be changed with 'platform' and 'refversion' options. The function returns a vector of probes that falls into the TSS region of the gene.

### Usage

```
getProbesByTSS(
  geneName,
  upstream = 1500,
  dwestream = 1500,
  platform = c("EPIC", "HM450"),
  refversion = c("hg38", "hg19")
)
```

### Arguments

geneName	gene name
upstream	the number of base pairs to expand upstream the TSS
dwestream	the number of base pairs to expand dwestream the TSS
platform	EPIC or HM450
refversion	hg38 or hg19

### Value

probes that fall into the given gene

### Examples

```
probes <- getProbesByTSS('CDKN2A')
```

---

getRefSet                              *Retrieve reference set*

---

### Description

The function retrieves the curated reference DNA methylation status for a set of cell type names under the Infinium platform. Supported cell types include "CD4T", "CD19B", "CD56NK", "CD14Monocytes", "granulocytes", "scFat", "skin" etc. See package sesameData for more details. The function output a matrix with probes on the rows and specified cell types on the columns. 0 suggests unmethylation and 1 suggests methylation. Intermediate methylation and nonclusive calls are left with NA.

**Usage**

```
getRefSet(cells = NULL, platform = c("EPIC", "HM450"))
```

**Arguments**

cells	reference cell types
platform	EPIC or HM450

**Value**

g, a 0/1 matrix with probes on the rows and specified cell types on the columns.

**Examples**

```
betas <- getRefSet('CD4T', platform='HM450')
```

---

getSegment	<i>Select segment from coefficient table</i>
------------	--

---

**Description**

This function takes a coefficient table and returns a subset of the table targeting only the specified segment using segment ID.

**Usage**

```
getSegment(cf1, seg.id)
```

**Arguments**

cf1	coefficient table of one factor from DMR
seg.id	segment ID

**Value**

coefficient table from given segment

**Examples**

```
data <- sesameDataGet('HM450.76.TCGA.matched')
cf <- DMR(data$betas, data$sampleInfo, ~type)
getSegment(cf[[1]], cf[[1]][['Seg.ID']][1])
```

---

getSexInfo	<i>Get sex-related information</i>
------------	------------------------------------

---

**Description**

The function takes a `SigSet` and returns a vector of three numerics: the median intensity of chrY probes; the median intensity of chrX probes; and fraction of intermediate chrX probes. chrX and chrY probes excludes pseudo-autosomal probes.

**Usage**

```
getSexInfo(sset)
```

**Arguments**

sset                    a `SigSet`

**Value**

medianY and medianX, fraction of XCI, methylated and unmethylated X probes, median intensities of auto-chromosomes.

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
getSexInfo(sset)
```

---

IG	<i>IG getter generic</i>
----	--------------------------

---

**Description**

IG getter generic  
Get IG slot of `SigSet` class

**Usage**

```
IG(x)

## S4 method for signature 'SigSet'
IG(x)
```

**Arguments**

x                    object of `SigSet`

**Value**

The IG slot of `SigSet`

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
head(IG(sset))
```

---

IG<- *IG replacement generic*

---

**Description**

IG replacement generic  
 Replace IG slot of SigSet class

**Usage**

```
IG(x) <- value

## S4 replacement method for signature 'SigSet'
IG(x) <- value
```

**Arguments**

x	object of SigSet
value	new value

**Value**

a new SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
df <- IG(sset)
df[1,1] <- 10
IG(sset) <- df
```

---

II *II getter generic*

---

**Description**

II getter generic  
 Get II slot of SigSet class

**Usage**

```
II(x)

## S4 method for signature 'SigSet'
II(x)
```

**Arguments**

x                    object of SigSet

**Value**

The II slot of SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
head(II(sset))
```

---

II<-                    *II replacement generic*

---

**Description**

II replacement generic  
Replace II slot of SigSet class

**Usage**

```
II(x) <- value

## S4 replacement method for signature 'SigSet'
II(x) <- value
```

**Arguments**

x                    object of SigSet  
value                new value

**Value**

a new SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
df <- II(sset)
df[1,1] <- 10
II(sset) <- df
```



---

inferEthnicity	<i>Infer Ethnicity</i>
----------------	------------------------

---

**Description**

This function uses both the built-in rsprobes as well as the type I Color-Channel-Switching probes to infer ethnicity.

**Usage**

```
inferEthnicity(sset)
```

**Arguments**

sset                    a SigSet

**Details**

sset better be background subtracted and dyebias corrected for best accuracy

Please note: the betas should come from sigset *\*without\** channel inference.

**Value**

string of ethnicity

**Examples**

```
sset <- makeExampleSeSAMEDataSet("HM450")
inferEthnicity(sset)
```

---

inferSex	<i>Infer Sex</i>
----------	------------------

---

**Description**

Infer Sex

**Usage**

```
inferSex(sset)
```

**Arguments**

sset                    a SigSet

**Value**

'F' or 'M' We established our sex calling based on the median intensity of chromosome X, Y and the fraction of intermediately methylated probes among the identified X-linked probes. This is similar to the approach by Minfi (Aryee et al., 2014) but also different in that we used the fraction of intermediate beta value rather than median intensity for all chromosome X probes. Instead of using all probes from the sex chromosomes, we used our curated set of Y chromosome probes and X-linked probes which exclude potential cross-hybridization and pseudo-autosomal effect.

XXY male (Klinefelter's), 45,X female (Turner's) can confuse the model sometimes. Our function works on a single sample.

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
inferSex(sset)
```

---

inferSexKaryotypes      *Infer Sex Karyotype*

---

**Description**

The function takes a SigSet and infers the sex chromosome Karyotype and presence/absence of X-chromosome inactivation (XCI). chrX, chrY and XCI are inferred relatively independently. This function gives a more detailed look of potential sex chromosome aberrations.

**Usage**

```
inferSexKaryotypes(sset)
```

**Arguments**

sset                      a SigSet

**Value**

Karyotype string, with XCI

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
inferSexKaryotypes(sset)
```

---

inferTypeIChannel	<i>Infer and reset color channel for Type-I probes instead of using what is specified in manifest. The results are stored to sset@extra\$IGG and sset@extra\$IRR slot.</i>
-------------------	--

---

**Description**

IGG => Type-I green that is inferred to be green IRR => Type-I red that is inferred to be red

**Usage**

```
inferTypeIChannel(
  sset,
  switch_failed = FALSE,
  verbose = FALSE,
  summary = FALSE
)
```

**Arguments**

sset	a SigSet
switch_failed	whether to switch failed probes (default to FALSE)
verbose	whether to print correction summary
summary	return summarized numbers only.

**Value**

a SigSet, or numerics if summary == TRUE

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
inferTypeIChannel(sset)
```

---

initFileSet	<i>initialize a fileSet class by allocating appropriate storage</i>
-------------	---

---

**Description**

initialize a fileSet class by allocating appropriate storage

**Usage**

```
initFileSet(map_path, platform, samples, probes = NULL, inc = 4)
```

**Arguments**

map_path	path of file to map
platform	EPIC, HM450 or HM27, consistent with sset@platform
samples	sample names
probes	probe names
inc	bytes per unit data storage

**Value**

a sesame::fileSet object

**Examples**

```
fset <- initFileSet('mybetas2', 'HM27', c('s1','s2'))
```

---

 IR

*IR getter generic*


---

**Description**

IR getter generic

Get IR slot of SigSet class

**Usage**

```
IR(x)
```

```
## S4 method for signature 'SigSet'
IR(x)
```

**Arguments**

x                    object of SigSet

**Value**

The IR slot of SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
head(IR(sset))
```

---

IR<- *IR replacement generic*

---

**Description**

IR replacement generic  
 Replace IR slot of SigSet class

**Usage**

```
IR(x) <- value

## S4 replacement method for signature 'SigSet'
IR(x) <- value
```

**Arguments**

x	object of SigSet
value	new value

**Value**

a new SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
df <- IR(sset)
df[1,1] <- 10
IR(sset) <- df
```

---

makeExampleSeSAMEDataSet  
*Make a simulated SeSAMEData set*

---

**Description**

Constructs a simulated SigSet dataset. For the given platform, randomly simulate methylated and unmethylated allele signals. In-band signals were simulated using a N(4000, 200) normal distribution. Out-of-band signals were simulated using a N(400, 200) normal distribution. Control signals were simulated using a N(400, 300) normal distribution.

**Usage**

```
makeExampleSeSAMEDataSet(platform = c("HM450", "EPIC", "HM27"))
```

**Arguments**

platform	optional, HM450, EPIC or HM27
----------	-------------------------------

**Value**

Object of class SigSet

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
```

---

```
makeExampleTinyEPICDataSet
```

*Make a tiny toy simulated EPIC data set*

---

**Description**

Construct a tiny EPIC SigSet of only 6 probes. In-band signals were simulated using a N(4000, 200) normal distribution. Out-of-band signals were simulated using a N(400, 200) normal distribution. Control signals were simulated using a N(400, 300) normal distribution.

**Usage**

```
makeExampleTinyEPICDataSet()
```

**Value**

Object of class SigSet

**Examples**

```
sset <- makeExampleTinyEPICDataSet()
```

---

```
mapFileSet
```

*Deposit data of one sample to a fileSet (and hence to file)*

---

**Description**

Deposit data of one sample to a fileSet (and hence to file)

**Usage**

```
mapFileSet(fset, sample, named_values)
```

**Arguments**

fset	a sesame::fileSet, as obtained via readFileSet
sample	sample name as a string
named_values	value vector named by probes

**Value**

a sesame::fileSet

**Examples**

```
## create two samples
fset <- initFileSet('mybetas2', 'HM27', c('s1','s2'))

## a hypothetical numeric array (can be beta values, intensities etc)
hypothetical <- setNames(runif(fset$n), fset$probes)

## map the numeric to file
mapFileSet(fset, 's1', hypothetical)

## get data
sliceFileSet(fset, 's1', 'cg00000292')
```

---

meanIntensity	<i>Mean Intensity</i>
---------------	-----------------------

---

**Description**

The function takes one single SigSet and computes mean intensity of all the in-band measurements. This includes all Type-I in-band measurements and all Type-II probe measurements. Both methylated and unmethylated alleles are considered. This function outputs a single numeric for the mean.

**Usage**

```
meanIntensity(sset)
```

**Arguments**

sset            a SigSet

**Value**

mean of all intensities

**Examples**

```
sset <- makeExampleSeSAMeDataSet()
meanIntensity(sset)
```

---

MValueToBetaValue	<i>Convert M-value to beta-value</i>
-------------------	--------------------------------------

---

**Description**

Convert M-value to beta-value (aka inverse logit transform)

**Usage**

```
MValueToBetaValue(m)
```

**Arguments**

m	a vector of M values
---	----------------------

**Value**

a vector of beta values

**Examples**

```
MValueToBetaValue(c(-3, 0, 3))
```

---

noob	<i>Noob background correction</i>
------	-----------------------------------

---

**Description**

The function takes a SigSet and returns a modified SigSet with background subtracted. Background was modelled in a normal distribution and true signal in an exponential distribution. The Norm-Exp deconvolution is parameterized using Out-Of-Band (oob) probes

**Usage**

```
noob(sset, oobRprobes = NULL, oobGprobes = NULL, offset = 15)
```

**Arguments**

sset	a SigSet
oobRprobes	out-of-band red probes, if not given use all oobR
oobGprobes	out-of-band grn probes, if not given use all oobG
offset	offset

**Value**

a new SigSet with noob background correction

**Examples**

```
sset <- makeExampleTinyEPICDataSet()
sset.nb <- noob(sset)
```



---

noobsb	<i>Background subtraction with bleeding-through subtraction</i>
--------	---

---

**Description**

The function takes a SigSet and returns a modified SigSet with background subtracted. Signal bleed-through was modelled using a linear model with error estimated from cross-channel regression. Norm-Exp deconvolution using Out-Of-Band (oob) probes.

**Usage**

```
noobsb(sset, offset = 15, detailed = FALSE)
```

**Arguments**

sset	a SigSet
offset	offset
detailed	if TRUE, return a list of SigSet and regression function

**Value**

a modified SigSet with background correction

**Examples**

```
sset <- makeExampleSeSAMEDataSet('HM450')
sset.nb <- noobsb(sset)
```

---

oobG	<i>oobG getter generic</i>
------	----------------------------

---

**Description**

oobG getter generic  
Get oobG slot of SigSet class

**Usage**

```
oobG(x)

## S4 method for signature 'SigSet'
oobG(x)
```

**Arguments**

x	object of SigSet
---	------------------

**Value**

The oobG slot of SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
head(oobG(sset))
```

---

```
oobG<-          oobG replacement generic
```

---

**Description**

oobG replacement generic  
 Replace oobG slot of SigSet class

**Usage**

```
oobG(x) <- value

## S4 replacement method for signature 'SigSet'
oobG(x) <- value
```

**Arguments**

x	object of SigSet
value	new value

**Value**

a new SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
df <- oobG(sset)
df[1,1] <- 10
oobG(sset) <- df
```

---

```
oobR          oobR getter generic
```

---

**Description**

oobR getter generic  
 Get oobR slot of SigSet class

**Usage**

```
oobR(x)

## S4 method for signature 'SigSet'
oobR(x)
```

**Arguments**

x                    object of SigSet

**Value**

The oobR slot of SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
head(oobR(sset))
```

---

oobR<-                    *oobR replacement generic*

---

**Description**

oobR replacement generic  
Replace oobR slot of SigSet class

**Usage**

```
oobR(x) <- value

## S4 replacement method for signature 'SigSet'
oobR(x) <- value
```

**Arguments**

x                    object of SigSet  
value                new value

**Value**

a new SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
df <- oobR(sset)
df[1,1] <- 10
oobR(sset) <- df
```

openSesame

*The openSesame pipeline*

---

**Description**

This function is a simple wrapper of noob + nonlinear dye bias correction + pOOBAH masking.

**Usage**

```
openSesame(  
  x,  
  platform = "",  
  manifest = NULL,  
  what = "beta",  
  BPPARAM = SerialParam(),  
  ...  
)
```

**Arguments**

x	SigSet(s), IDAT prefix(es), minfi GenomicRatioSet(s), or RGChannelSet(s)
platform	optional platform string
manifest	optional dynamic manifest
what	either 'sigset' or 'beta'
BPPARAM	get parallel with MulticoreParam(n)
...	parameters to getBetas

**Details**

If the input is an IDAT prefix or a SigSet, the output is the beta value numerics. If the input is a minfi GenomicRatioSet or RGChannelSet, the output is the sesamized GenomicRatioSet.

**Value**

a numeric vector for processed beta values

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset  
IDATprefixes <- searchIDATprefixes(  
  system.file("extdata", "", package = "sesameData"))  
betas <- openSesame(IDATprefixes)
```

---

openSesameToFile      *openSesame pipeline with file-backed storage*

---

**Description**

openSesame pipeline with file-backed storage

**Usage**

```
openSesameToFile(map_path, idat_dir, BPPARAM = SerialParam(), inc = 4)
```

**Arguments**

map_path	path of file to be mapped (beta values file)
idat_dir	source IDAT directory
BPPARAM	get parallel with MulticoreParam(2)
inc	bytes per item data storage. increase to 8 if precision is important. Most cases 32-bit representation is enough.

**Value**

a sesame::fileSet

**Examples**

```
openSesameToFile('mybetas',
  system.file('extdata', package='sesameData'))
```

---

parseGEOSignalABFile      *Parse GEO signal-A/B File into a SigSet List*

---

**Description**

This function is meant to be a convenience function for parsing data from Signal\_A and Signal\_B file provided by GEO. In many cases, this function generates a "partial" SigSet due to lack of out-of-band signal and control probe measurement in those Signal\_A/B files. The detection p-value is based on a fixed normal distribution rather than from negative control or OOB probes.

**Usage**

```
parseGEOSignalABFile(path, platform = "HM450", drop = TRUE, parallel = TRUE)
```

**Arguments**

path	path to Signal-A/B file downloaded from GEO. The file can remain gzipped.
platform	HM450, EPIC or HM27
drop	whether to reduce to SigSet when there is only one sample.
parallel	whether to use multiple cores.

**Value**

a SigSetList or a SigSet

**Examples**

```
path = system.file(
  'extdata',
  'GSE36369_NonEBV_SignalA_SignalB_3samples_1k.txt.gz',
  package='sesame')
ssets <- parseGEOSignalABFile(path)
```

---

predictAgeHorvath353    *Horvath 353 age predictor*

---

**Description**

The function takes a named numeric vector of beta values. The name attribute contains the probe ID (cg, ch or rs IDs). The function looks for overlapping probes and estimate age using Horvath aging model (Horvath 2013 Genome Biology). The function outputs a single numeric of age in years.

**Usage**

```
predictAgeHorvath353(betas)
```

**Arguments**

betas                    a probeID-named vector of beta values

**Value**

age in years

**Examples**

```
betas <- sesameDataGet('HM450.1.TCGA.PAAD')$betas
predictAgeHorvath353(betas)
```

---

predictAgePheno            *Phenotypic age predictor*

---

**Description**

The function takes a named numeric vector of beta values. The name attribute contains the probe ID (cg, ch or rs IDs). The function looks for overlapping probes and estimate age using Horvath aging model (Levine et al. 2018 Aging, 513 probes). The function outputs a single numeric of age in years.

**Usage**

```
predictAgePheno(betas)
```

**Arguments**

betas                    a probeID-named vector of beta values

**Value**

age in years

**Examples**

```
betas <- sesameDataGet('HM450.1.TCGA.PAAD')$betas  
predictAgePheno(betas)
```

---

predictAgeSkinBlood    *Horvath Skin and Blood age predictor*

---

**Description**

The function takes a named numeric vector of beta values. The name attribute contains the probe ID (cg, ch or rs IDs). The function looks for overlapping probes and estimate age using Horvath aging model (Horvath et al. 2018 Aging, 391 probes). The function outputs a single numeric of age in years.

**Usage**

```
predictAgeSkinBlood(betas)
```

**Arguments**

betas                    a probeID-named vector of beta values

**Value**

age in years

**Examples**

```
betas <- sesameDataGet('HM450.1.TCGA.PAAD')$betas  
predictAgeSkinBlood(betas)
```

---

print.fileSet	<i>Print a fileSet</i>
---------------	------------------------

---

**Description**

Print a fileSet

**Usage**

```
## S3 method for class 'fileSet'  
print(x, ...)
```

**Arguments**

x	a sesame::fileSet
...	stuff for print

**Value**

string representation

**Examples**

```
fset <- initFileSet('mybetas2', 'HM27', c('s1','s2'))  
fset
```

---

print.sesameQC	<i>Print sesameQC object</i>
----------------	------------------------------

---

**Description**

Print sesameQC object

**Usage**

```
## S3 method for class 'sesameQC'  
print(x, ...)
```

**Arguments**

x	a sesameQC object
...	extra parameter for print

**Value**

print sesameQC result on screen



**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
sesameQC(sset)
```

---

probeNames	<i>Get Probe Names of SigSet class</i>
------------	--

---

**Description**

Get Probe Names of SigSet class

**Usage**

```
probeNames(x)

## S4 method for signature 'SigSet'
probeNames(x)
```

**Arguments**

x                    object of Sigset

**Value**

A char vector

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset
head(probeNames(sset))
```

---

pval	<i>pval getter generic</i>
------	----------------------------

---

**Description**

pval getter generic  
Get pval slot of SigSet class

**Usage**

```
pval(x)

## S4 method for signature 'SigSet'
pval(x)
```

**Arguments**

x                    object of SigSet

**Value**

The pval slot of SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset  
head(pval(sset))
```

---

pval<-                      *pval replacement generic*

---

**Description**

pval replacement generic

Replace pval slot of SigSet class

**Usage**

```
pval(x) <- value  
  
## S4 replacement method for signature 'SigSet'  
pval(x) <- value
```

**Arguments**

x	object of SigSet
value	new value

**Value**

a new SigSet

**Examples**

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset  
df <- pval(sset)  
df[1] <- 0.01  
pval(sset) <- df
```

---

qualityMask	<i>Mask beta values by design quality</i>
-------------	---

---

**Description**

Currently quality masking only supports three platforms

**Usage**

```
qualityMask(sset, mask.use.tcga = FALSE)
```

**Arguments**

sset	a SigSet object
mask.use.tcga	whether to use TCGA masking, only applies to HM450

**Value**

a filtered SigSet

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
sset.masked <- qualityMask(sset)
```

---

qualityRank	<i>This function looks at public data of similar nature e.g., tissue, FFPE vs non-FFPE, etc to evaluate the quality of the target data quality</i>
-------------	--

---

**Description**

This function looks at public data of similar nature e.g., tissue, FFPE vs non-FFPE, etc to evaluate the quality of the target data quality

**Usage**

```
qualityRank(sset, tissue = NULL, samplePrep = NULL, raw = FALSE)
```

**Arguments**

sset	a raw (unprocessed) SigSet
tissue	A string (blood,buccal and saliva)
samplePrep	FFPE, fresh, frozen
raw	to return the raw comparison table

**Value**

three numbers: 1. The number of public samples compared. 2. The fraction of public samples with more nondetection, and 3. The fraction of public samples with lower mean intensity 4. The higher the fraction, the better the sample.

## Examples

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset  
ranks <- qualityRank(sset)
```

---

readFileSet	<i>Read an existing fileSet from storage</i>
-------------	--

---

## Description

This function only reads the meta-data.

## Usage

```
readFileSet(map_path)
```

## Arguments

map\_path            path of file to map (should contain valid \_idx.rds index)

## Value

a sesame::fileSet object

## Examples

```
## create two samples  
fset <- initFileSet('mybetas2', 'HM27', c('s1','s2'))  
  
## a hypothetical numeric array (can be beta values, intensities etc)  
hypothetical <- setNames(runif(fset$n), fset$probes)  
  
## map the numeric to file  
mapFileSet(fset, 's1', hypothetical)  
  
## read it from file  
fset <- readFileSet('mybetas2')  
  
## get data  
sliceFileSet(fset, 's1', 'cg00000292')
```

---

readIDATpair	<i>Import a pair of IDATs from one sample</i>
--------------	---

---

**Description**

The function takes a prefix string that are shared with `_Grn.idat` and `_Red.idat`. The function returns a `SigSet`.

**Usage**

```
readIDATpair(
  prefix.path,
  platform = "",
  manifest = NULL,
  controls = NULL,
  verbose = FALSE
)
```

**Arguments**

<code>prefix.path</code>	sample prefix without <code>_Grn.idat</code> and <code>_Red.idat</code>
<code>platform</code>	EPIC, HM450 and HM27 etc.
<code>manifest</code>	optional design manifest file
<code>controls</code>	optional control probe manifest file
<code>verbose</code>	be verbose? (FALSE)

**Value**

a `SigSet`

**Examples**

```
sset <- readIDATpair(sub('_Grn.idat', '', system.file(
  "extdata", "4207113116_A_Grn.idat", package = "sesameData")))
```

---

reIdentify	<i>Re-identify IDATs by restoring scrambled SNP intensities</i>
------------	---

---

**Description**

This requires setting a seed with a secret number that was used to de-identify the IDAT (see example). This requires a secret number that was used to de-identify the IDAT

**Usage**

```
reIdentify(path, out_path = NULL, snps = NULL, mft = NULL)
```

**Arguments**

path	input IDAT file
out_path	output IDAT file
snps	SNP definition, if not given, default to SNP probes
mft	sesame-compatible manifest if non-standard

**Value**

NULL, changes made to the IDAT files

**Examples**

```
temp_out <- tempfile("test")

set.seed(123)
reIdentify(system.file(
  "extdata", "4207113116_A_Grn.idat", package = "sesameData"), temp_out)
unlink(temp_out)
```

---

reopenSesame	<i>re-compute beta value for GenomicRatioSet</i>
--------------	--

---

**Description**

re-compute beta value for GenomicRatioSet

**Usage**

```
reopenSesame(x, naFrac = 0.2)
```

**Arguments**

x	GenomicRatioSet
naFrac	maximum NA fraction for a probe before it gets dropped (1)

**Value**

a GenomicRatioSet

---

RGChannelSetToSigSets *Convert RGChannelSet (minfi) to a list of SigSet (SeSAME)*

---

### Description

Notice the colData() and rowData() is lost. Most cases, rowData is empty anyway.

### Usage

```
RGChannelSetToSigSets(rgSet, manifest = NULL, BPPARAM = SerialParam())
```

### Arguments

rgSet	a minfi::RGChannelSet
manifest	manifest file
BPPARAM	get parallel with MulticoreParam(n)

### Value

a list of sesame::SigSet

### Examples

```
if (require(FlowSorted.Blood.450k)) {
  rgSet <- FlowSorted.Blood.450k[,1:2]
  ssets <- RGChannelSetToSigSets(rgSet)
}
```

---

searchIDATprefixes *Identify IDATs from a directory*

---

### Description

The input is the directory name as a string. The function identifies all the IDAT files under the directory. The function returns a vector of such IDAT prefixes under the directory.

### Usage

```
searchIDATprefixes(dir.name, recursive = TRUE, use.basename = TRUE)
```

### Arguments

dir.name	the directory containing the IDAT files.
recursive	search IDAT files recursively
use.basename	basename of each IDAT path is used as sample name This won't work in rare situation where there are duplicate IDAT files.

**Value**

the IDAT prefixes (a vector of character strings).

**Examples**

```
## only search what are directly under
IDATprefixes <- searchIDATprefixes(
  system.file("extdata", "", package = "sesameData"))

## search files recursively is by default
IDATprefixes <- searchIDATprefixes(
  system.file(package = "sesameData"), recursive=TRUE)
```

---

segmentBins	<i>Segment bins using DNACopy</i>
-------------	-----------------------------------

---

**Description**

Segment bins using DNACopy

**Usage**

```
segmentBins(bin.signals, bin.coords)
```

**Arguments**

bin.signals	bin signals (input)
bin.coords	bin coordinates

**Value**

segment signal data frame

---

sesameQC	<i>Generate summary numbers that indicative of experiment quality Please provide a raw sigset (before any preprocessing). Usually directly from readIDATpair</i>
----------	--

---

**Description**

Generate summary numbers that indicative of experiment quality Please provide a raw sigset (before any preprocessing). Usually directly from readIDATpair

**Usage**

```
sesameQC(sset)
```

**Arguments**

sset	a SigSet object
------	-----------------



**Value**

a sesameQC class object

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
sesameQC(sset)
```

---

sesamize	<i>"fix" an RGChannelSet (for which IDATs may be unavailable) with Sesame The input is an RGSet and the output is a sesamized minfi::GenomicRatioSet</i>
----------	--

---

**Description**

"fix" an RGChannelSet (for which IDATs may be unavailable) with Sesame The input is an RGSet and the output is a sesamized minfi::GenomicRatioSet

**Usage**

```
sesamize(
  rgSet,
  naFrac = 1,
  BPPARAM = SerialParam(),
  HDF5 = NULL,
  HDF5SEdestination = paste0(tempdir(check = TRUE), "/sesamize_HDF5_scratch"),
  replace = FALSE
)
```

**Arguments**

rgSet	an RGChannelSet, perhaps with colData of various flavors
naFrac	maximum NA fraction for a probe before it gets dropped (1)
BPPARAM	get parallel with MulticoreParam(n)
HDF5	is the rgSet HDF5-backed? if so, avoid eating RAM (perhaps)
HDF5SEdestination	character(1) path to where the HDF5-backed GenomicRatioSet will be stored
replace	logical(1) passed to saveHDF5SummarizedExperiment

**Value**

a sesamized GenomicRatioSet

**Note**

We employ BPRED0 for a second chance if bplapply hits an error.

---

show, SigSet-method      *The display method for SigSet*

---

### Description

The function outputs the number of probes in each category and the first few signal measurements.

### Usage

```
## S4 method for signature 'SigSet'
show(object)
```

### Arguments

object                    displayed object

### Value

message of number of probes in each category.

### Examples

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
print(sset)
```

---

signalMU                    *report M and U for regular probes*

---

### Description

report M and U for regular probes

### Usage

```
signalMU(sset)
```

### Arguments

sset                      a SigSet

### Value

a data frame of M and U columns

### Examples

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
signalMU(sset)
```

SigSet-class

*SigSet class***Description**

This is the main data class for SeSAmE. The class holds different classes of signal intensities.

The function takes a string describing the platform of the data. It can be one of "HM27", "HM450" or "EPIC".

The function takes a string describing the platform of the data. It can be one of "HM27", "HM450" or "EPIC".

**Usage**

```
## S4 method for signature 'SigSet'
initialize(.Object, platform, ...)
```

```
SigSet(...)
```

**Arguments**

.Object	target object
platform	"EPIC", "HM450", "HM27" or other strings for custom arrays
...	additional arguments

**Value**

a SigSet object

a SigSet object

**Slots**

IG intensity table for type I probes in green channel

IR intensity table for type I probes in red channel

II intensity table for type II probes

oobG out-of-band probes in green channel

oobR out-of-band probes in red channel

ctl all the control probe intensities

pval named numeric vector of p-values

extra extra data, currently, IGG => Type-I green that is inferred to be green IRR => Type-I red that is inferred to be red pvals => list of other pvals

platform "EPIC", "HM450" or "HM27"

**Examples**

```
## Create an empty EPIC object.
SigSet("EPIC")
SigSet('EPIC')
```

---

SigSetList                    *constructor*

---

**Description**

constructor

**Usage**

```
SigSetList(...)
```

**Arguments**

...                    the SigSet objects that will be the List elements

**Value**

a SigSetList

**Examples**

```
sset1 <- readIDATpair(file.path(system.file(
  'extdata', '', package='sesameData'), '4207113116_A'))

sset2 <- readIDATpair(file.path(system.file(
  'extdata', '', package='sesameData'), '4207113116_B'))

SigSetList(sset1, sset2)
```

---

SigSetList-class            *a List of SigSets with some methods of its own*

---

**Description**

a List of SigSets with some methods of its own

---

SigSetList-methods        *SigSetList methods (centralized). Currently scarce... 'show' print a summary of the SigSetList.*

---

**Description**

SigSetList methods (centralized). Currently scarce...  
'show' print a summary of the SigSetList.

**Usage**

```
## S4 method for signature 'SigSetList'
show(object)
```

**Arguments**

object            a SigSetList

**Value**

Description of SigSetList

**Examples**

```
SigSetListFromPath(system.file("extdata", "", package = "sesameData"))
```

---

SigSetListFromIDATs    *read IDATs into a SigSetList*

---

**Description**

FIXME: switch from 'parallel' to BiocParallel

**Usage**

```
SigSetListFromIDATs(stubs, parallel = FALSE)
```

**Arguments**

stubs            the IDAT filename stubs  
parallel        run in parallel? (default FALSE)

**Value**

a SigSetList

**Examples**

```
## a SigSetList of length 1
ssets <- SigSetListFromIDATs(file.path(
  system.file("extdata", "", package = "sesameData"), "4207113116_A"))
```

---

SigSetListFromPath    *read an entire directory's worth of IDATs into a SigSetList*

---

**Description**

read an entire directory's worth of IDATs into a SigSetList

**Usage**

```
SigSetListFromPath(path = ".", parallel = FALSE, recursive = TRUE)
```

**Arguments**

path	the path from which to read IDATs (default ".")
parallel	run in parallel? (default FALSE)
recursive	whether to search recursively

**Value**

a SigSetList

**Examples**

```
## Load all IDATs from directory
ssets <- SigSetListFromPath(
  system.file("extdata", "", package = "sesameData"))
```

---

SigSetsToRGChannelSet *Convert sesame::SigSet to minfi::RGChannelSet*

---

**Description**

Convert sesame::SigSet to minfi::RGChannelSet

**Usage**

```
SigSetsToRGChannelSet(ssets, BPPARAM = SerialParam(), annotation = NA)
```

**Arguments**

ssets	a list of sesame::SigSet
BPPARAM	get parallel with MulticoreParam(n)
annotation	the minfi annotation string, guessed if not given

**Value**

a minfi::RGChannelSet

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
rgSet <- SigSetsToRGChannelSet(sset)
```

---

SigSetToRatioSet	<i>Convert one sesame::SigSet to minfi::RatioSet</i>
------------------	--

---

**Description**

Convert one sesame::SigSet to minfi::RatioSet

**Usage**

```
SigSetToRatioSet(sset, annotation = NA)
```

**Arguments**

sset	a sesame::SigSet
annotation	minfi annotation string

**Value**

a minfi::RatioSet

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset
ratioSet <- SigSetToRatioSet(sset)
```

---

sliceFileSet	<i>Slice a fileSet with samples and probes</i>
--------------	--

---

**Description**

Slice a fileSet with samples and probes

**Usage**

```
sliceFileSet(fset, samples = fset$samples, probes = fset$probes, memmax = 10^5)
```

**Arguments**

fset	a sesame::fileSet, as obtained via readFileSet
samples	samples to query (default to all samples)
probes	probes to query (default to all probes)
memmax	maximum items to read from file to memory, to protect from accidental memory congestion.

**Value**

a numeric matrix of length(samples) columns and length(probes) rows

**Examples**

```
## create two samples
fset <- initFileSet('mybetas2', 'HM27', c('s1','s2'))

## a hypothetical numeric array (can be beta values, intensities etc)
hypothetical <- setNames(runif(fset$n), fset$probes)

## map the numeric to file
mapFileSet(fset, 's1', hypothetical)

## get data
sliceFileSet(fset, 's1', 'cg00000292')
```

---

SNPcheck	<i>Check sample identity using SNP probes</i>
----------	---

---

**Description**

Check sample identity using SNP probes

**Usage**

```
SNPcheck(betas)
```

**Arguments**

betas                    numeric matrix (row: probes, column: samples)

**Value**

grid object plotting SNP clustering

**Examples**

```
betas <- sesameDataGet('HM450.10.TCGA.PAAD.normal')
SNPcheck(betas)
```

---

subsetSignal	<i>Select a subset of probes</i>
--------------	----------------------------------

---

**Description**

The function takes a SigSet as input and output another SigSet with probes from the given probe selection.

**Usage**

```
subsetSignal(sset, probes)
```



**Arguments**

sset                    a SigSet  
probes                  target probes

**Value**

another sset with probes specified

**Examples**

```
sset <- sesameDataGet('EPIC.1.LNCaP')$sset  
subsetSignal(sset, rownames(slot(sset, 'IR')))
```

---

topLoci

*Top loci in differential methylation*

---

**Description**

This is a convenience function to show top differential methylated segments. The function takes a coefficient table as input and output the same table ordered by the significance of the locus.

**Usage**

```
topLoci(cf1)
```

**Arguments**

cf1                    coefficient table of one factor from diffMeth

**Value**

coefficient table ordered by p-value of each locus

**Examples**

```
data <- sesameDataGet('HM450.76.TCGA.matched')  
cf <- DMR(data$betas, data$sampleInfo, ~type)  
topLoci(cf[[1]])
```

---

topSegments	<i>Top segments in differential methylation</i>
-------------	---

---

**Description**

This is a utility function to show top differential methylated segments. The function takes a coefficient table as input and output the same table ordered by the significance of the segments.

**Usage**

```
topSegments(cf1)
```

**Arguments**

cf1                    coefficient table of one factor from DMR

**Value**

coefficient table ordered by adjusted p-value of segments

**Examples**

```
data <- sesameDataGet('HM450.76.TCGA.matched')
cf <- DMR(data$betas, data$sampleInfo, ~type)
topSegments(cf[[1]])
```

---

totalIntensities	<i>M+U Intensities for All Probes</i>
------------------	---------------------------------------

---

**Description**

The function takes one single SigSet and computes total intensity of all the in-band measurements by summing methylated and unmethylated alleles. This function outputs a single numeric for the mean.

**Usage**

```
totalIntensities(sset)
```

**Arguments**

sset                    a SigSet

**Value**

a vector of M+U signal for each probe

**Examples**

```
sset <- makeExampleSeSAMeDataSet()
intensities <- totalIntensities(sset)
```

---

totalIntensityZscore    *Calculate intensity Z-score*

---

### Description

This function compute intensity Z-score with respect to the mean. Log10 transformation is done first. Probes of each design type are grouped before Z-scores are computed.

### Usage

```
totalIntensityZscore(sset)
```

### Arguments

sset                    a SigSet

### Value

a vector of Z-score for each probe

### Examples

```
sset <- sesameDataGet('HM450.1.TCGA.PAAD')$sset  
head(totalIntensityZscore(sset))
```

---

twoCompsEst2                    *Estimate the fraction of the 2nd component in a 2-component mixture*

---

### Description

Estimate the fraction of the 2nd component in a 2-component mixture

### Usage

```
twoCompsEst2(  
  pop1,  
  pop2,  
  target,  
  use.ave = TRUE,  
  diff_1m2u = NULL,  
  diff_1u2m = NULL  
)
```

**Arguments**

pop1	Reference methylation level matrix for population 1
pop2	Reference methylation level matrix for population 2
target	Target methylation level matrix to be analyzed
use.ave	use population average in selecting differentially methylated probes
diff_1m2u	A vector of differentially methylated probes (methylated in population 1 but unmethylated in population 2)
diff_1u2m	A vector of differentially methylated probes (unmethylated in population 1 but methylated in population 2)

**Value**

Estimate of the 2nd component in the 2-component mixture

---

visualizeGene	<i>Visualize Gene</i>
---------------	-----------------------

---

**Description**

Visualize the beta value in heatmaps for a given gene. The function takes a gene name which is taken from the UCSC refGene. It searches all the transcripts for the given gene and optionally extend the span by certain number of base pairs. The function also takes a beta value matrix with sample names on the columns and probe names on the rows. The function can also work on different genome builds (default to hg38, can be hg19).

**Usage**

```
visualizeGene(
  geneName,
  betas,
  platform = c("EPIC", "HM450"),
  upstream = 2000,
  dwnstream = 2000,
  refversion = c("hg38", "hg19"),
  ...
)
```

**Arguments**

geneName	gene name
betas	beta value matrix (row: probes, column: samples)
platform	HM450 or EPIC (default)
upstream	distance to extend upstream
dwnstream	distance to extend downstream
refversion	hg19 or hg38 (default)
...	additional options, see visualizeRegion

**Value**

None

**Examples**

```
betas <- sesameDataGet('HM450.76.TCGA.matched')$betas
visualizeGene('ADA', betas, 'HM450')
```

---

visualizeProbes	<i>Visualize Region that Contains the Specified Probes</i>
-----------------	--

---

**Description**

Visualize the beta value in heatmaps for the genomic region containing specified probes. The function works only if specified probes can be spanned by a single genomic region. The region can cover more probes than specified. Hence the plotting heatmap may encompass more probes. The function takes as input a string vector of probe IDs (cg/ch/rs-numbers). if draw is FALSE, the function returns the subset beta value matrix otherwise it returns the grid graphics object.

**Usage**

```
visualizeProbes(
  probeNames,
  betas,
  platform = c("EPIC", "HM450"),
  refversion = c("hg38", "hg19"),
  upstream = 1000,
  dwestream = 1000,
  ...
)
```

**Arguments**

probeNames	probe names
betas	beta value matrix (row: probes, column: samples)
platform	HM450 or EPIC (default)
refversion	hg19 or hg38 (default)
upstream	distance to extend upstream
dwestream	distance to extend downstream
...	additional options, see visualizeRegion

**Value**

None

**Examples**

```
betas <- sesameDataGet('HM450.76.TCGA.matched')$betas
visualizeProbes(c('cg22316575', 'cg16084772', 'cg20622019'), betas, 'HM450')
```

---

visualizeRegion	<i>Visualize Region</i>
-----------------	-------------------------

---

### Description

The function takes a genomic coordinate (chromosome, start and end) and a beta value matrix (probes on the row and samples on the column). It plots the beta values as a heatmap for all probes falling into the genomic region. If 'draw=TRUE' the function returns the plotted grid graphics object. Otherwise, the selected beta value matrix is returned. 'cluster.samples=TRUE/FALSE' controls whether hierarchical clustering is applied to the subset beta value matrix.

### Usage

```
visualizeRegion(
  chr,
  plt.beg,
  plt.end,
  betas,
  platform = c("EPIC", "HM450"),
  refversion = c("hg38", "hg19"),
  sample.name.fontsize = 10,
  heat.height = NULL,
  draw = TRUE,
  show.sampleNames = TRUE,
  show.samples.n = NULL,
  show.probeNames = TRUE,
  cluster.samples = FALSE,
  nprobes.max = 1000,
  na.rm = FALSE,
  dmin = 0,
  dmax = 1
)
```

### Arguments

chr	chromosome
plt.beg	begin of the region
plt.end	end of the region
betas	beta value matrix (row: probes, column: samples)
platform	EPIC or HM450
refversion	hg38 or hg19
sample.name.fontsize	sample name font size
heat.height	heatmap height (auto inferred based on rows)
draw	draw figure or return betas
show.sampleNames	whether to show sample names
show.samples.n	number of samples to show (default: all)

show.probeNames	whether to show probe names
cluster.samples	whether to cluster samples
nprobes.max	maximum number of probes to plot
na.rm	remove probes with all NA.
dmin	data min
dmax	data max

**Value**

graphics or a matrix containing the captured beta values

**Examples**

```
betas <- sesameDataGet('HM450.76.TCGA.matched')$betas
visualizeRegion('chr20', 44648623, 44652152, betas, 'HM450')
```

---

visualizeSegments	<i>Visualize segments</i>
-------------------	---------------------------

---

**Description**

The function takes a CNSegment object obtained from cnSegmentation and plot the bin signals and segments (as horizontal lines).

**Usage**

```
visualizeSegments(seg, to.plot = NULL)
```

**Arguments**

seg	a CNSegment object
to.plot	chromosome to plot (by default plot all chromosomes)

**Details**

require ggplot2, scales

**Value**

plot graphics

**Examples**

```
seg <- sesameDataGet('EPIC.1.LNCaP')$seg
visualizeSegments(seg)
```

# Index

- \* **DNAMethylation**
  - sesame-package, 4
- \* **Microarray**
  - sesame-package, 4
- \* **QualityControl**
  - sesame-package, 4
- \_PACKAGE (sesame-package), 4
- as.data.frame.sesameQC, 5
- BetaValueToMValue, 5
- binSignals, 6
- bisConversionControl, 6
- buildControlMatrix450k, 7
- chipAddressToSignal, 7
- cnSegmentation, 8
- createUCSCtrack, 9
- ctl, 9
- ctl, SigSet-method (ctl), 9
- ctl<-, 10
- ctl<-, SigSet-method (ctl<-), 10
- deIdentify, 11
- detectionMask, 11
- detectionPfixedNorm, 12
- detectionPnegEcdf, 13
- detectionPnegNorm, 13
- detectionPnegNormGS, 14
- detectionPnegNormTotal, 15
- detectionPoobEcdf, 15
- detectionZero, 16
- diffRefSet, 17
- DML, 17
- DMR, 18
- dyeBiasCorr, 19
- dyeBiasCorrMostBalanced, 20
- dyeBiasCorrTypeINorm, 20
- estimateCellComposition, 21
- estimateLeukocyte, 21
- extra, 22
- extra, SigSet-method (extra), 22
- extra<-, 23
- extra<-, SigSet-method (extra<-), 23
- formatVCF, 23
- getAFTTypeIbySumAlleles, 24
- getBetas, 25
- getBinCoordinates, 25
- getNormCtls, 26
- getProbesByGene, 26
- getProbesByRegion, 27
- getProbesByTSS, 28
- getRefSet, 28
- getSegment, 29
- getSexInfo, 30
- IG, 30
- IG, SigSet-method (IG), 30
- IG<-, 31
- IG<-, SigSet-method (IG<-), 31
- II, 31
- II, SigSet-method (II), 31
- II<-, 32
- II<-, SigSet-method (II<-), 32
- inferEthnicity, 33
- inferSex, 33
- inferSexKaryotypes, 34
- inferTypeIChannel, 35
- initFileSet, 35
- initialize, SigSet-method (SigSet-class), 59
- IR, 36
- IR, SigSet-method (IR), 36
- IR<-, 37
- IR<-, SigSet-method (IR<-), 37
- makeExampleSeSAMEDataSet, 37
- makeExampleTinyEPICDataSet, 38
- mapFileSet, 38
- meanIntensity, 39
- MValueToBetaValue, 40
- noob, 40
- noobsb, 41
- oobG, 41
- oobG, SigSet-method (oobG), 41
- oobG<-, 42



- oobG<- , SigSet-method (oobG<-), 42
- oobR, 42
- oobR, SigSet-method (oobR), 42
- oobR<- , 43
- oobR<- , SigSet-method (oobR<-), 43
- openSesame, 44
- openSesameToFile, 45
  
- parseGEOSignalABFile, 45
- pOOBAH (detectionPoobEcdf), 15
- predictAgeHorvath353, 46
- predictAgePheno, 46
- predictAgeSkinBlood, 47
- print.fileSet, 48
- print.sesameQC, 48
- probeNames, 49
- probeNames, SigSet-method (probeNames), 49
- pval, 49
- pval, SigSet-method (pval), 49
- pval<- , 50
- pval<- , SigSet-method (pval<-), 50
  
- qualityMask, 51
- qualityRank, 51
  
- readFileSet, 52
- readIDATpair, 53
- reIdentify, 53
- reopenSesame, 54
- RGChannelSetToSigSets, 55
  
- searchIDATprefixes, 55
- segmentBins, 56
- sesame (sesame-package), 4
- sesame-package, 4
- sesameQC, 56
- sesamize, 57
- show, SigSet-method, 58
- show, SigSetList-method (SigSetList-methods), 60
- signalMU, 58
- SigSet (SigSet-class), 59
- SigSet-class, 59
- SigSetList, 60
- SigSetList-class, 60
- SigSetList-methods, 60
- SigSetListFromIDATs, 61
- SigSetListFromPath, 61
- SigSetsToRGChannelSet, 62
- SigSetToRatioSet, 63
- sliceFileSet, 63
- SNPcheck, 64
  
- subsetSignal, 64
  
- topLoci, 65
- topSegments, 66
- totalIntensities, 66
- totalIntensityZscore, 67
- twoCompsEst2, 67
  
- visualizeGene, 68
- visualizeProbes, 69
- visualizeRegion, 70
- visualizeSegments, 71