

# Package ‘sesame’

April 16, 2019

**Type** Package

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

**Description** Tools For analyzing Illumina Infinium DNA methylation arrays.

**Version** 1.0.0

**Depends** R (>= 3.5), sesameData, methods

**License** Artistic-2.0

**RoxygenNote** 6.1.0

**Imports** R6, grDevices, utils, illuminaio, MASS, GenomicRanges, IRanges, grid, preprocessCore, stats, S4Vectors, randomForest, wheatmap, ggplot2, parallel, DNACopy

**Suggests** scales, knitr, rmarkdown, testthat, minfi, SummarizedExperiment, FlowSorted.CordBloodNorway.450k

**VignetteBuilder** knitr

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

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

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

**Collate** 'SigSetList.R' 'sesame.R' 'SigSetMethods.R' 'age.R' 'background\_correction.R' 'cell\_composition.R' 'cnv.R' 'controls.R' 'detection.R' 'dye\_bias.R' 'open.R' 'sesamize.R' 'simulate\_data.R' 'snp.R' 'utils.R' 'visualize.R'

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

**git\_branch** RELEASE\_3\_8

**git\_last\_commit** 8a9c43d

**git\_last\_commit\_date** 2018-10-30

**Date/Publication** 2019-04-15

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

**Maintainer** Wanding Zhou <[zhouwanding@gmail.com](mailto:zhouwanding@gmail.com)>

**R topics documented:**

|                                      |    |
|--------------------------------------|----|
| sesame-package . . . . .             | 3  |
| BetaValueToMValue . . . . .          | 4  |
| binSignals . . . . .                 | 4  |
| bisConversionControl . . . . .       | 5  |
| buildControlMatrix450k . . . . .     | 5  |
| chipAddressToSignal . . . . .        | 6  |
| cnSegmentation . . . . .             | 6  |
| ctl . . . . .                        | 7  |
| ctl<- . . . . .                      | 8  |
| detectionPnegEcdf . . . . .          | 8  |
| detectionPnegNorm . . . . .          | 9  |
| detectionPnegNormGS . . . . .        | 9  |
| detectionPnegNormTotal . . . . .     | 10 |
| detectionPoobEcdf . . . . .          | 10 |
| diffRefSet . . . . .                 | 11 |
| dyeBiasCorr . . . . .                | 12 |
| dyeBiasCorrMostBalanced . . . . .    | 12 |
| dyeBiasCorrTypeINorm . . . . .       | 13 |
| estimateCellComposition . . . . .    | 13 |
| estimateLeukocyte . . . . .          | 14 |
| getAFTypeIbySumAlleles . . . . .     | 15 |
| getBetas . . . . .                   | 15 |
| getBetasTypeIbySumChannels . . . . . | 16 |
| getBinCoordinates . . . . .          | 17 |
| getNormCtls . . . . .                | 17 |
| getProbesByGene . . . . .            | 18 |
| getProbesByRegion . . . . .          | 18 |
| getProbesByTSS . . . . .             | 19 |
| getRefSet . . . . .                  | 20 |
| getSexInfo . . . . .                 | 20 |
| IG . . . . .                         | 21 |
| IG<- . . . . .                       | 21 |
| II . . . . .                         | 22 |
| II<- . . . . .                       | 23 |
| inferEthnicity . . . . .             | 23 |
| inferSex . . . . .                   | 24 |
| inferSexKaryotypes . . . . .         | 25 |
| IR . . . . .                         | 25 |
| IR<- . . . . .                       | 26 |
| makeExampleSeSAMEDataSet . . . . .   | 26 |
| makeExampleTinyEPICDataSet . . . . . | 27 |
| meanIntensity . . . . .              | 27 |
| MValueToBetaValue . . . . .          | 28 |
| noob . . . . .                       | 28 |
| noobsb . . . . .                     | 29 |
| oobG . . . . .                       | 30 |
| oobG<- . . . . .                     | 30 |
| oobR . . . . .                       | 31 |
| oobR<- . . . . .                     | 31 |
| openSesame . . . . .                 | 32 |

|                                |    |
|--------------------------------|----|
| predictAgeHorvath353 . . . . . | 33 |
| predictAgePheno . . . . .      | 33 |
| predictAgeSkinBlood . . . . .  | 34 |
| pval . . . . .                 | 34 |
| pval<- . . . . .               | 35 |
| readIDATpair . . . . .         | 35 |
| reopenSesame . . . . .         | 36 |
| searchIDATprefixes . . . . .   | 36 |
| segmentBins . . . . .          | 37 |
| sesamize . . . . .             | 37 |
| show,SigSet-method . . . . .   | 38 |
| signalR6toS4 . . . . .         | 38 |
| SignalSet . . . . .            | 39 |
| SigSet-class . . . . .         | 40 |
| SigSetList . . . . .           | 41 |
| SigSetList-class . . . . .     | 41 |
| SigSetList-methods . . . . .   | 41 |
| SigSetListFromIDATs . . . . .  | 42 |
| SigSetListFromPath . . . . .   | 42 |
| SNPcheck . . . . .             | 43 |
| subsetSignal . . . . .         | 44 |
| totalIntensities . . . . .     | 44 |
| totalIntensityZscore . . . . . | 45 |
| visualizeGene . . . . .        | 45 |
| visualizeProbes . . . . .      | 46 |
| visualizeRegion . . . . .      | 47 |
| visualizeSegments . . . . .    | 48 |

## Index 49

---

|                |                                     |
|----------------|-------------------------------------|
| 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 TCGA standard pipeline
betas <- getBetas(dyeBiasCorrTypeINorm(noob(sset)))
```

---

|                   |                                      |
|-------------------|--------------------------------------|
| BetaValueToMValue | <i>Convert beta-value to M-value</i> |
|-------------------|--------------------------------------|

---

**Description**

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

**Usage**

```
BetaValueToMValue(b)
```

**Arguments**

b                      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 *Compute internal bisulfite conversion control*

---

**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. The lower the GCT score, the more likely over-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)
```

**Arguments**

dm                      data frame in chip address, 2 columns: cy3/Grn and cy5/Red

**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             *Perform copy number segmentation*

---

**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)
```

---

ctl                    *ctl getter generic*

---

**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
```

---

```
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)
```

**Arguments**

|      |          |
|------|----------|
| sset | a SigSet |
|------|----------|

**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)
```

**Arguments**

sset                    a SigSet

**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)
```

**Arguments**

sset                    a SigSet

**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.

**Usage**

```
detectionPnegNormTotal(sset)
```

**Arguments**

```
sset          a SigSet
```

**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 (Pvals by Out-Of-Band Array Hybridization)

**Usage**

```
detectionPoobEcdf(sset)
```

**Arguments**

```
sset          a SigSet
```

**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)
```

---

|            |  |
|------------|--|
| 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'))
```

---

`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**

|                   |                        |
|-------------------|------------------------|
| <code>sset</code> | a SigSet               |
| <code>ref</code>  | 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**

|                    |                              |
|--------------------|------------------------------|
| <code>ssets</code> | 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**

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

**Value**

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

---

|                                |  |
|--------------------------------|--|
| <code>estimateLeukocyte</code> | <i>Estimate leukocyte fraction using a two-component model</i> |
|--------------------------------|--|

---

**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**

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

**Value**

leukocyte estimate, a numeric vector

**Examples**

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

---

 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

*Get beta Values*


---

### Description

Get beta Values

### Usage

```
getBetas(sset, quality.mask = TRUE, nondetection.mask = TRUE,
  mask.use.tcga = FALSE, pval.threshold = 0.05)
```

### Arguments

sset                    SigSet  
 quality.mask    whether to mask low quality probes  
 nondetection.mask    whether to mask nondetection  
 mask.use.tcga    whether to use TCGA masking, only applies to HM450  
 pval.threshold    p-value threshold for nondetection mask

**Value**

a numeric vector, beta values

**Examples**

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

---

getBetasTypeIbySumChannels

*Get beta values treating type I by summing channels*

---

**Description**

This function 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**

```
getBetasTypeIbySumChannels(sset, quality.mask = TRUE,
  nondetection.mask = TRUE, pval.threshold = 0.05)
```

**Arguments**

|                   |   |
|-------------------|---|
| sset              | SigSet                                  |
| quality.mask      | whether to mask low quality probes      |
| nondetection.mask | whether to mask nondetection            |
| pval.threshold    | p-value threshold for nondetection mask |

**Value**

beta values

**Examples**

```
sset <- makeExampleSeSAMEDataSet()
betas <- getBetasTypeIbySumChannels(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

---

|             |   |
|-------------|---|
| getNormCtls | <i>get normalization control signal</i> |
|-------------|---|

---

**Description**

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

**Usage**

```
getNormCtls(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.ctl <- getNormCtls(sset)
```

---

getProbesByGene      *Get Probes by Gene*

---

### 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,
  dwstream = 0, refversion = c("hg38", "hg19"))
```

### Arguments

|            |   |
|------------|---|
| geneName   | gene name   |
| platform   | EPIC or HM450                                       |
| upstream   | number of bases to expand upstream of target gene   |
| dwstream   | 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, dwstream=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**

|            |                                |
|------------|--------------------------------|
| chrn       | 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 | <i>Get Probes by Gene Transcription Start Site (TSS)</i> |
|----------------|--|

---

**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, dstream = 1500,
  platform = c("EPIC", "HM450"), refversion = c("hg38", "hg19"))
```

**Arguments**

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

**Value**

probes that fall into the given gene

**Examples**

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

---

|           |                               |
|-----------|-------------------------------|
| getRefSet | <i>Retrieve reference set</i> |
|-----------|-------------------------------|

---

### 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')
```

---

|            |                                    |
|------------|------------------------------------|
| 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 <code>SigSet</code> |
|------|-----------------------|

### Value

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

**Examples**

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

---

IG *IG getter generic*

---

**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                    *Infer Ethnicity*

---

**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

**Value**

string of ethnicity

**Examples**

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

---

inferSex

*Infer Sex*

---

**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 | <i>Infer Sex Karyotype</i> |
|--------------------|----------------------------|

---

**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)
```

---

|    |                          |
|----|--------------------------|
| IR | <i>IR getter generic</i> |
|----|--------------------------|

---

**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()
```

---

```
meanIntensity
```

*Mean Intensity*

---

**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            *Convert M-value to beta-value*

---

**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                        *Noob background correction*

---

**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, offset = 15)
```

**Arguments**

sset            a SigSet  
offset         offset

**Value**

a new SigSet with noob background correction

**Examples**

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

---

noobsb                            *Background subtraction with bleeding-through subtraction*

---

**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<- | <i>oobG replacement generic</i> |
|--------|---------------------------------|

---

**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 | <i>oobR getter generic</i> |
|------|----------------------------|

---

**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<- | <i>oobR replacement generic</i> |
|--------|---------------------------------|

---

**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, ...)
```

**Arguments**

|     |  |
|-----|--|
| x   | SigSet(s), IDAT prefix(es), minfi GenomicRatioSet(s), or RGChannelSet(s) |
| ... | 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)
```



---

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)
```

---

pval                    *pval getter generic*

---

**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<- *Replace pval slot of SigSet class*

---

**Description**

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
```

---

readIDATpair *Import a pair of IDATs from one sample*

---

**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, verbose = FALSE)
```

**Arguments**

|             |   |
|-------------|---|
| prefix.path | sample prefix without <code>_Grn.idat</code> and <code>_Red.idat</code> |
| verbose     | be verbose? (FALSE)   |

**Value**

a SigSet

**Examples**

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

---

|              |  |
|--------------|--|
| 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

---

|                    |  |
|--------------------|--|
| searchIDATprefixes | <i>Identify IDATs from a directory</i> |
|--------------------|--|

---

**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 = FALSE)
```

**Arguments**

|           |  |
|-----------|--|
| dir.name  | the directory containing the IDAT files. |
| recursive | search IDAT files recursively            |

**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
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

---

|          |  |
|----------|--|
| sesamize | <i>"fix" an RGset (for which IDATs may be unavailable) with Sesame</i> |
|----------|--|

---

**Description**

"fix" an RGset (for which IDATs may be unavailable) with Sesame

**Usage**

```
sesamize(x, naFrac = 1, parallel = FALSE)
```

**Arguments**

|          |   |
|----------|---|
| x        | an RGChannelSet, perhaps with colData of various flavors    |
| naFrac   | maximum NA fraction for a probe before it gets dropped (1)  |
| parallel | attempt to run in parallel? (This is a bad idea on laptops) |

**Value**

a sesamized GenomicRatioSet from the input RGChannelSet

**Examples**

```
# Takes about two minutes to process 48 samples on my 48-core desktop
if (require(FlowSorted.CordBloodNorway.450k)) {
  sesamized <- sesamize(
    FlowSorted.CordBloodNorway.450k[,1:2])
}
```

---

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)
```

---

signalR6toS4                      *sesame R6 to S4*

---

**Description**

SignalSet-SigSet conversion

**Usage**

```
signalR6toS4(sset)
```

**Arguments**

sset                              signalset in R6

**Value**

signalset in S4

**Examples**

```
sset <- SignalSet$new('EPIC')
sset$IG <- matrix(1:4, nrow=2)
sset$IR <- matrix(1:4, nrow=2)
sset$II <- matrix(1:4, nrow=2)
sset$oobG <- matrix(1:4, nrow=2)
sset$oobR <- matrix(1:4, nrow=2)
sset$ctl <- data.frame(G=1:2,R=3:4)
sset$pval <- rep(0,2)

signalR6toS4(sset)
```

---

SignalSet

*SignalSet class*

---

**Description**

SignalSet class

**Usage**

SignalSet

**Format**

An [R6Class](#) object.

**Value**

Object of class SignalSet

**Fields**

platform platform name, supports "EPIC", "HM450" and "HM27"  
 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  
 mask probe mask

**Documentation** For full documentation of each method go to  
 new(platform) Create a SignalSet in the specified platform  
 detectPValue() Detect P-value for each probe  
 toM() Convert to M values  
 totalIntensities() Total intensity on each probe

**Examples**

```
SignalSet$new("EPIC")
```

---

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 = c("EPIC", "HM450",
  "HM27"), ...)

SigSet(...)
```

**Arguments**

|          |                           |
|----------|---------------------------|
| .Object  | target object             |
| platform | "EPIC", "HM450" or "HM27" |
| ...      | 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  
 ct1 all the control probe intensities  
 pval named numeric vector of p-values  
 platform "EPIC", "HM450" or "HM27"

**Examples**

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



---

|            |                    |
|------------|--------------------|
| SigSetList | <i>constructor</i> |
|------------|--------------------|

---

**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 | <i>a List of SigSets with some methods of its own</i> |
|------------------|---|

---

**Description**

a List of SigSets with some methods of its own

---

|                    |  |
|--------------------|--|
| SigSetList-methods | <i>SigSetList methods (centralized). Currently scarce... 'show' print a summary of the SigSetList.</i> |
|--------------------|--|

---

**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"))
```

---

SNPcheck

*Check sample identity using SNP probes*

---

**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                      *Select a subset of probes*

---

### 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')))
```

---

totalIntensities                      *M+U Intensities for All Probes*

---

### 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))
```

---

visualizeGene            *Visualize Gene*

---

### 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, dstream = 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                      |
| dstream    | 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

*Visualize Region that Contains the Specified Probes*


---

**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, dwnstream = 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                      |
| dwnstream  | 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(chrm, 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, na.rm = FALSE, dmin = 0, dmax = 1)
```

### Arguments

|                      |  |
|----------------------|--|
| chrm                 | 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                       |
| 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

- \*Topic **DNAMethylation**
  - sesame-package, 3
- \*Topic **Microarray**
  - sesame-package, 3
- \*Topic **QualityControl**
  - sesame-package, 3
- \*Topic **datasets**
  - SignalSet, 39
- \_PACKAGE (sesame-package), 3
- BetaValueToMValue, 4
- binSignals, 4
- bisConversionControl, 5
- buildControlMatrix450k, 5
- chipAddressToSignal, 6
- cnSegmentation, 6
- ctl, 7
- ctl, SigSet-method (ctl), 7
- ctl<-, 8
- ctl<-, SigSet-method (ctl<-), 8
- detectionPnegEcdf, 8
- detectionPnegNorm, 9
- detectionPnegNormGS, 9
- detectionPnegNormTotal, 10
- detectionPoobEcdf, 10
- diffRefSet, 11
- dyeBiasCorr, 12
- dyeBiasCorrMostBalanced, 12
- dyeBiasCorrTypeINorm, 13
- estimateCellComposition, 13
- estimateLeukocyte, 14
- getAFTYPEIbySumAlleles, 15
- getBetas, 15
- getBetasTypeIbySumChannels, 16
- getBinCoordinates, 17
- getNormCtls, 17
- getProbesByGene, 18
- getProbesByRegion, 18
- getProbesByTSS, 19
- getRefSet, 20
- getSexInfo, 20
- IG, 21
- IG, SigSet-method (IG), 21
- IG<-, 21
- IG<-, SigSet-method (IG<-), 21
- II, 22
- II, SigSet-method (II), 22
- II<-, 23
- II<-, SigSet-method (II<-), 23
- inferEthnicity, 23
- inferSex, 24
- inferSexKaryotypes, 25
- initialize, SigSet-method (SigSet-class), 40
- IR, 25
- IR, SigSet-method (IR), 25
- IR<-, 26
- IR<-, SigSet-method (IR<-), 26
- makeExampleSeSAMEDataSet, 26
- makeExampleTinyEPICDataSet, 27
- meanIntensity, 27
- MValueToBetaValue, 28
- noob, 28
- noobsb, 29
- oobG, 30
- oobG, SigSet-method (oobG), 30
- oobG<-, 30
- oobG<-, SigSet-method (oobG<-), 30
- oobR, 31
- oobR, SigSet-method (oobR), 31
- oobR<-, 31
- oobR<-, SigSet-method (oobR<-), 31
- openSesame, 32
- predictAgeHorvath353, 33
- predictAgePheno, 33
- predictAgeSkinBlood, 34
- pval, 34
- pval, SigSet-method (pval), 34
- pval<-, 35
- pval<-, SigSet-method (pval<-), 35
- R6Class, 39

- readIDATpair, [35](#)
- reopenSesame, [36](#)
  
- searchIDATprefixes, [36](#)
- segmentBins, [37](#)
- sesame (sesame-package), [3](#)
- sesame-package, [3](#)
- sesamize, [37](#)
- show, SigSet-method, [38](#)
- show, SigSetList-method  
(SigSetList-methods), [41](#)
- signalR6toS4, [38](#)
- SignalSet, [39](#)
- SigSet (SigSet-class), [40](#)
- SigSet-class, [40](#)
- SigSetList, [41](#)
- SigSetList-class, [41](#)
- SigSetList-methods, [41](#)
- SigSetListFromIDATs, [42](#)
- SigSetListFromPath, [42](#)
- SNPcheck, [43](#)
- subsetSignal, [44](#)
  
- totalIntensities, [44](#)
- totalIntensityZscore, [45](#)
  
- visualizeGene, [45](#)
- visualizeProbes, [46](#)
- visualizeRegion, [47](#)
- visualizeSegments, [48](#)