

# Package ‘sesameData’

April 16, 2024

**Type** Package

**Title** Supporting Data for SeSAmE Package

**Description** Provides supporting annotation and test data for SeSAmE package. This includes chip tango addresses, mapping information, performance annotation, and trained predictor for Infinium array data. This package provides user access to essential annotation data for working with many generations of the Infinium DNA methylation array. Current we support human array (HM27, HM450, EPIC), mouse array (MM285) and the Horvath-MethylChip40 (Mammal40) array.

**Version** 1.20.0

**License** Artistic-2.0

**Depends** R (>= 4.2.0), ExperimentHub, AnnotationHub

**Imports** utils, readr, stringr, GenomicRanges, S4Vectors, IRanges, GenomeInfoDb

**Suggests** BiocGenerics, sesame, testthat, knitr, rmarkdown

**biocViews** ExperimentData, MicroarrayData, Genome, ExperimentHub, MethylationArrayData

**VignetteBuilder** knitr

**NeedsCompilation** no

**RoxygenNote** 7.2.3

**Encoding** UTF-8

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

**git\_branch** RELEASE\_3\_18

**git\_last\_commit** ff46e9d

**git\_last\_commit\_date** 2023-10-24

**Repository** Bioconductor 3.18

**Date/Publication** 2024-04-16

**Author** Wanding Zhou [aut, cre] (<<https://orcid.org/0000-0001-9126-1932>>),  
Hui Shen [aut],  
Timothy Triche [ctb]

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

**R topics documented:**

build_GENCODE_gtf . . . . .	2
df_master . . . . .	3
extend . . . . .	3
inferPlatformFromProbeIDs . . . . .	4
sesameDataCache . . . . .	4
sesameDataCacheAll . . . . .	5
sesameDataCacheExample . . . . .	5
sesameDataGet . . . . .	6
sesameDataGet_checkEnv . . . . .	6
sesameDataGet_resetEnv . . . . .	7
sesameDataHas . . . . .	7
sesameDataList . . . . .	8
sesameData_annoProbes . . . . .	8
sesameData_check_genome . . . . .	9
sesameData_check_platform . . . . .	10
sesameData_getAutosomeProbes . . . . .	10
sesameData_getGenesByProbes . . . . .	11
sesameData_getGenomeInfo . . . . .	12
sesameData_getManifestGRanges . . . . .	12
sesameData_getProbesByChromosome . . . . .	13
sesameData_getProbesByGene . . . . .	14
sesameData_getProbesByRegion . . . . .	15
sesameData_getProbesByTSS . . . . .	16
sesameData_getTxnGRanges . . . . .	17
sesameData_txnToGeneGRanges . . . . .	17
<b>Index</b>	<b>19</b>

---

build_GENCODE_gtf	<i>build GENCODE gtf</i>
-------------------	--------------------------

---

**Description**

build GENCODE gtf

**Usage**

```
build_GENCODE_gtf(x)
```

**Arguments**

x                    GENCODE ftp url

**Value**

GRangesList

---

df_master	<i>Master data frame for all object to cache</i>
-----------	--

---

**Description**

This is an internal object which will be updated on every new release library(ExperimentHub) eh <- query(ExperimentHub(localHub=FALSE), c("sesameData", "v1.13.1")) data.frame(name=eh\$title, eh=names(eh))

**Value**

master sheet of sesameData objects

---

extend	<i>Extend a GRanges</i>
--------	-------------------------

---

**Description**

source: <https://support.bioconductor.org/p/78652/>

**Usage**

```
extend(gr, upstream = 0, downstream = 0)
```

**Arguments**

gr	a GenomicRanges::GRanges
upstream	distance to expand upstream
downstream	distance to expand downstream

**Value**

a GenomicRanges::GRanges

**Examples**

```
library(GenomicRanges)
extend(GRanges("chr1", IRanges(10,20), "-"), upstream = 5, downstream = 3)
```

---

```
inferPlatformFromProbeIDs  
    infer platform from Probe_IDs
```

---

**Description**

infer platform from Probe\_IDs

**Usage**

```
inferPlatformFromProbeIDs(Probe_IDs, silent = FALSE)
```

**Arguments**

Probe_IDs	probe IDs
silent	suppress message

**Value**

a platform code

**Examples**

```
inferPlatformFromProbeIDs(c("cg14620903", "cg22464003"))
```

---

```
sesameDataCache    Cache all SeSAmE data
```

---

**Description**

Cache all SeSAmE data

**Usage**

```
sesameDataCache()
```

**Value**

TRUE

**Examples**

```
if(FALSE) { sesameDataCacheAll() }
```

---

sesameDataCacheAll      *Cache all SeSAmE data*

---

**Description**

Cache all SeSAmE data

**Usage**

```
sesameDataCacheAll()
```

**Value**

TRUE

**Examples**

```
if(FALSE) { sesameDataCacheAll() }
```

---

sesameDataCacheExample  
*Cache SeSAmE data for specific platform*

---

**Description**

Cache SeSAmE data for specific platform

**Usage**

```
sesameDataCacheExample()
```

**Value**

TRUE

**Examples**

```
if(FALSE) { sesameDataCacheExample() }
```

sesameDataGet            *Get SeSAmE data*

---

**Description**

Get SeSAmE data

**Usage**

```
sesameDataGet(title, verbose = FALSE)
```

**Arguments**

title	title of the data
verbose	whether to output ExperimentHub message

**Value**

data object

**Examples**

```
sesameDataCacheExample()  
EPIC.1.SigDF <- sesameDataGet('EPIC.1.SigDF')
```

---

sesameDataGet\_checkEnv  
*Check whether the title exists in cacheEnv*

---

**Description**

Check whether the title exists in cacheEnv

**Usage**

```
sesameDataGet_checkEnv(title)
```

**Arguments**

title	the title to check
-------	--------------------

**Value**

the data associated with the title or NULL if title doesn't exist

---

`sesameDataGet_resetEnv`*Empty cache environment to free memory*

---

**Description**

When this function is called `sesameDataGet` will retrieve all data from disk again instead of using the in-memory cache, i.e., `sesameData:::cacheEnv`.

**Usage**

```
sesameDataGet_resetEnv()
```

**Details**

Note this is different from `sesameDataClearHub` which empties the `ExperimentHub` on disk.

**Value**

`gc()` output

**Examples**

```
sesameDataGet_resetEnv()
```

---

`sesameDataHas`*Whether sesameData has*

---

**Description**

Whether `sesameData` has

**Usage**

```
sesameDataHas(data_titles)
```

**Arguments**

`data_titles` data titles to check

**Value**

a boolean vector the same length as `data_titles`

**Examples**

```
sesameDataHas(c("EPIC.address", "EPIC.address.Nonexist"))
```

---

sesameDataList	<i>List all SeSAmE data</i>
----------------	-----------------------------

---

**Description**

List all SeSAmE data

**Usage**

```
sesameDataList(filter = NULL, full = FALSE)
```

**Arguments**

filter	keyword to filter title, optional
full	whether to display all columns

**Value**

all titles from SeSAmE Data

**Examples**

```
sesameDataList("KYCG")
```

---

sesameData_annoProbes	<i>Annotate Probes by Probe ID</i>
-----------------------	------------------------------------

---

**Description**

Please note that if unfound, the annotation will be NA. The probe will always be kept in the output.

**Usage**

```
sesameData_annoProbes(  
  Probe_IDs,  
  regs = NULL,  
  collapse = TRUE,  
  chooseOne = FALSE,  
  column = NULL,  
  sep = ",",  
  out_name = NULL,  
  platform = NULL,  
  genome = NULL,  
  silent = FALSE  
)
```



**Arguments**

Probe_IDs	a character vector of probe IDs
regs	a GenomicRanges::GRanges object against which probes will be annotated, default to genes if not given
collapse	whether to collapse multiple regs into one
chooseOne	choose an arbitrary annotation if multiple exist default to FALSE. which concatenates all with ", "
column	which column in regs to annotate
sep	the delimiter for collapsing
out_name	column header of the annotation, use column if not given
platform	EPIC, MM285 etc. will infer from Probe_IDs if not given
genome	hg38, mm10, ... will infer if not given. For additional mapping, download the GRanges object from <a href="http://zwdzwd.github.io/InfiniumAnnotation">http://zwdzwd.github.io/InfiniumAnnotation</a> and provide the following argument ..., genome = sesameAnno_buildManifestGRanges("downloaded_file"),... to this function.
silent	suppress messages

**Value**

a GRanges with annotated column If a probe has no overlap with regs, it will be included in the results with NA. But if a probe is not included in the manifest (due to mappability), it won't be included in the results.

**Examples**

```
library(GenomicRanges)
regs = sesameData_getTxnGRanges("mm10")
Probe_IDs = names(sesameData_getManifestGRanges("MM285"))
anno = sesameData_annoProbes(Probe_IDs, promoters(regs), column="gene_name")
```

---

sesameData\_check\_genome

*check genome supported for a platform*

---

**Description**

check genome supported for a platform

**Usage**

```
sesameData_check_genome(genome, platform)
```

**Arguments**

genome	mm10, hg38, ..., or NULL
platform	HM27, HM450, EPIC, ...

**Value**

genome as string

**Examples**

```
sesameData_check_genome(NULL, "Mamma140")
```

---

```
sesameData_check_platform  
    check platform code
```

---

**Description**

check platform code

**Usage**

```
sesameData_check_platform(platform = NULL, probes = NULL)
```

**Arguments**

platform	input platform
probes	probes by which the platform may be guessed

**Value**

platform code

**Examples**

```
sesameData_check_platform("HM450")
```

---

```
sesameData_getAutosomeProbes  
    Get autosome probes
```

---

**Description**

Get autosome probes

**Usage**

```
sesameData_getAutosomeProbes(platform = NULL, genome = NULL)
```

**Arguments**

platform 'EPIC', 'HM450' etc.  
 genome hg38, mm10, ... will infer if not given. For additional mapping, download the GRanges object from <http://zwdzwd.github.io/InfiniumAnnotation> and provide the following argument ..., genome = sesameAnno\_buildManifestGRanges("downloaded\_file"),... to this function.

**Value**

GRanges of autosome probes

**Examples**

```
auto_probes <- sesameData_getAutosomeProbes('Mammal40')
```

---

sesameData\_getGenesByProbes  
*get genes next to certain probes*

---

**Description**

get genes next to certain probes

**Usage**

```
sesameData_getGenesByProbes(  
  Probe_IDs,  
  platform = NULL,  
  genome = NULL,  
  max_distance = 10000  
)
```

**Arguments**

Probe\_IDs probe IDs  
 platform EPIC, HM450, ... will infer if not given  
 genome hg38, mm10, ... will infer if not given. For additional mapping, download the GRanges object from <http://zwdzwd.github.io/InfiniumAnnotation> and provide the following argument ..., genome = sesameAnno\_buildManifestGRanges("downloaded\_file"),... to this function.  
 max\_distance maximum distance to gene (default: 10000)

**Value**

a GRanges object for overlapping genes

**Examples**

```
sesameData_getGenesByProbes(c("cg14620903", "cg22464003"))
```

---

```
sesameData_getGenomeInfo
```

*Get genome info files*

---

**Description**

Get genome info files

**Usage**

```
sesameData_getGenomeInfo(genome)
```

**Arguments**

genome                   hg38, mm10, or GRanges with a metadata(genome)[["genome"]]

**Value**

a list of genome info files

**Examples**

```
ginfo <- sesameData_getGenomeInfo("hg38")
```

---

```
sesameData_getManifestGRanges
```

*get Infinium manifest GRanges*

---

**Description**

Note that some unaligned probes are not included. For full manifest, please visit <http://zwdzwd.github.io/InfiniumAnnotation>

**Usage**

```
sesameData_getManifestGRanges(platform, genome = NULL)
```

**Arguments**

platform               Mammal40, MM285, EPIC, and HM450

genome                 hg38, mm10, ... will infer if not given. For additional mapping, download the GRanges object from <http://zwdzwd.github.io/InfiniumAnnotation> and provide the following argument ..., genome = sesameAnno\_buildManifestGRanges("downloaded\_file"),... to this function.

**Value**

GRanges

**Examples**

```
gr <- sesameData_getManifestGRanges("Mamma140")
```

---

sesameData\_getProbesByChromosome  
*Get Probes by Chromosome*

---

**Description**

Get Probes by Chromosome

**Usage**

```
sesameData_getProbesByChromosome(chrms, platform = NULL, genome = NULL)
```

**Arguments**

chrms	chromosomes to subset
platform	EPIC, HM450, Mouse
genome	hg38, mm10, ... will infer if not given. For additional mapping, download the GRanges object from <a href="http://zwdzwd.github.io/InfiniumAnnotation">http://zwdzwd.github.io/InfiniumAnnotation</a> and provide the following argument ..., genome = sesameAnno_buildManifestGRanges("downloaded_file"),... to this function.

**Value**

GRanges of selected probes

**Examples**

```
Xprobes <- sesameData_getProbesByChromosome('chrX', "Mamma140")
```

---

`sesameData_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 'genome' options. The function returns a vector of probes that falls into the given gene.

**Usage**

```
sesameData_getProbesByGene(  
  gene_name = NULL,  
  platform = NULL,  
  upstream = 0,  
  downstream = 0,  
  genome = NULL  
)
```

**Arguments**

gene_name	gene name, if NULL return all genes
platform	EPIC or HM450
upstream	number of bases to expand upstream of target gene
downstream	number of bases to expand downstream of target gene
genome	hg38 or hg19

**Value**

GRanges containing probes that fall into the given gene

**Examples**

```
probes <- sesameData_getProbesByGene(  
  'DNMT3A', "Mamma140", upstream=500, downstream=500)
```

---

`sesameData_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

```
sesameData_getProbesByRegion(  
  regs,  
  chrm = NULL,  
  beg = 1,  
  end = -1,  
  platform = NULL,  
  genome = NULL  
)
```

## Arguments

<code>regs</code>	GRanges
<code>chrm</code>	chromosome, when given regs are ignored
<code>beg</code>	begin, 1 if omitted
<code>end</code>	end, chromosome end if omitted
<code>platform</code>	EPIC, HM450, ...
<code>genome</code>	hg38, mm10, ... will infer if not given. For additional mapping, download the GRanges object from <a href="http://zwdzwd.github.io/InfiniumAnnotation">http://zwdzwd.github.io/InfiniumAnnotation</a> and provide the following argument ..., <code>genome = sesameAnno_buildManifestGRanges("downloaded_file"),...</code> to this function.

## Value

GRanges of selected probes

## Examples

```
library(GenomicRanges)  
sesameData_getProbesByRegion(  
  GRanges('chr5', IRanges(135313937, 135419936)), platform = 'Mammal40')
```

---

`sesameData_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 'genome' options. The function returns a vector of probes that falls into the TSS region of the gene.

### Usage

```
sesameData_getProbesByTSS(  
  gene_name = NULL,  
  platform = NULL,  
  upstream = 1500,  
  downstream = 1500,  
  genome = NULL  
)
```

### Arguments

gene_name	gene name, if NULL, return all TSS probes
platform	EPIC, HM450, or MM285
upstream	the number of base pairs to expand upstream the TSS
downstream	the number of base pairs to expand downstream the TSS
genome	hg38, hg19 or mm10

### Value

probes that fall into the given gene

### Examples

```
probes <- sesameData_getProbesByTSS('DNMT3A', "Mamma140")
```



---

```
sesameData_getTxnGRanges
```

*convert GRangesList to transcript GRanges*

---

**Description**

convert GRangesList to transcript GRanges

**Usage**

```
sesameData_getTxnGRanges(genome = NULL, grl = NULL)
```

**Arguments**

genome	hg38, mm10, ...
grl	GRangesList object

**Value**

a GRanges object

**Examples**

```
txns <- sesameData_getTxnGRanges("mm10")  
## get verified protein-coding  
txns <- txns[(txns$transcript_type == "protein_coding" & txns$level <= 2)]
```

---

```
sesameData_txnToGeneGRanges
```

*convert transcript GRanges to gene GRanges*

---

**Description**

convert transcript GRanges to gene GRanges

**Usage**

```
sesameData_txnToGeneGRanges(txns)
```

**Arguments**

txns	GRanges object
------	----------------

**Value**

a GRanges object

**Examples**

```
txns <- sesameData_getTxnGRanges("mm10")  
genes <- sesameData_txnToGeneGRanges(txns)
```

# Index

[build\\_GENCODE\\_gtf](#), [2](#)

[df\\_master](#), [3](#)

[extend](#), [3](#)

[inferPlatformFromProbeIDs](#), [4](#)

[sesameData\\_annoProbes](#), [8](#)

[sesameData\\_check\\_genome](#), [9](#)

[sesameData\\_check\\_platform](#), [10](#)

[sesameData\\_getAutosomeProbes](#), [10](#)

[sesameData\\_getGenesByProbes](#), [11](#)

[sesameData\\_getGenomeInfo](#), [12](#)

[sesameData\\_getManifestGRanges](#), [12](#)

[sesameData\\_getProbesByChromosome](#), [13](#)

[sesameData\\_getProbesByGene](#), [14](#)

[sesameData\\_getProbesByRegion](#), [15](#)

[sesameData\\_getProbesByTSS](#), [16](#)

[sesameData\\_getTxnGRanges](#), [17](#)

[sesameData\\_txnToGeneGRanges](#), [17](#)

[sesameDataCache](#), [4](#)

[sesameDataCacheAll](#), [5](#)

[sesameDataCacheExample](#), [5](#)

[sesameDataGet](#), [6](#)

[sesameDataGet\\_checkEnv](#), [6](#)

[sesameDataGet\\_resetEnv](#), [7](#)

[sesameDataHas](#), [7](#)

[sesameDataList](#), [8](#)