

# Package ‘sesameData’

February 17, 2026

**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** 1.28.0

**License** Artistic-2.0

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

**Imports** utils, readr, stringr, GenomicRanges (>= 1.61.1), S4Vectors, IRanges, Seqinfo

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

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

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

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

**VignetteBuilder** knitr

**NeedsCompilation** no

**RoxygenNote** 7.3.2

**Encoding** UTF-8

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

**git\_branch** RELEASE\_3\_22

**git\_last\_commit** 851657b

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.22

**Date/Publication** 2026-02-17

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

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

## Contents

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

**Details**

Cache location is default to `/Users/zhouw3/Library/Caches/org.R-project.R/R/ExperimentHub/`

**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

---

```
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**

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

---

```
sesameDataCache      Cache SeSAmE data
```

---

**Description**

Cache SeSAmE data

**Usage**

```
sesameDataCache(data_titles = NULL)
```

**Arguments**

data_titles	data to cache, if not given will cache all
-------------	--

**Value**

TRUE

**Examples**

```
sesameDataCache("genomeInfo.hg38")
```

---

sesameDataCacheAll	<i>Cache all SeSAmE data</i>
--------------------	------------------------------

---

**Description**

Cache all SeSAmE data

**Usage**

```
sesameDataCacheAll(data_titles = NULL)
```

**Arguments**

data\_titles      data to cache, if not given will cache all

**Value**

TRUE

**Examples**

```
sesameDataCache("genomeInfo.hg38")
```

---

sesameDataGet	<i>Get SeSAmE data</i>
---------------	------------------------

---

**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**

```
sesameDataCache("EPIC.1.SigDF")  
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	<i>Whether sesameData has</i>
---------------	-------------------------------

---

**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 *Annotate Probes by Probe ID*

---

### Description

Columns in the manifests will be added to the annotation. 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 = ",",
  return_ov_probes = FALSE,
  return_ov_features = FALSE,
  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, if not given return all overlapping probes
sep	the delimiter for collapsing
return_ov_probes	if TRUE will return overlapping probes in a GRanges object.
return_ov_features	if TRUE will return overlapping features in a GRanges object.
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)
sesameDataCache(c(
  "genomeInfo.mm10", "MM285.address",
  "genomeInfo.hg38", "Mammal140.address"))

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

## get all genes associated with a probe set
genes = sesameData_getTxnGRanges("hg38", merge2gene = TRUE)
anno = sesameData_annoProbes(
  c("cg14620903", "cg22464003"), genes, return_ov_features=TRUE)
```

---

sesameData\_check\_genome

*Find genome assembly version(s) supported for a platform*

---

**Description**

Find genome assembly version(s) supported for a platform

**Usage**

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

**Arguments**

genome	mm10, hg38, ..., or NULL
platform	HM27, HM450, EPIC, EPICv2, MSA, ...

**Value**

genome as string

**Examples**

```
sesameData_check_genome(NULL, "Mammal140")
```

sesameData\_check\_platform  
*Check platform code*

---

**Description**

Note: custome platforms lead to error here.

**Usage**

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

**Arguments**

platform	input platform
probes	probes by which the platform may be guessed
silent	suppress message

**Value**

platform code

**Examples**

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

---

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**

```
sesameDataCache("genomeInfo.hg38")  
res <- 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 <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

### Examples

```
sesameDataCache("Mammal40.address")
res <- sesameData_getManifestGRanges("Mammal40")
```

---

```
sesameData_getProbesByGene
      Get Probes by Genes or Gene Promoters
```

---

### 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,
  promoter = FALSE,
  upstream = 1500,
  downstream = 1500,
  genome = NULL
)
```

**Arguments**

gene_name	gene name, if NULL return all genes
platform	EPIC or HM450
promoter	if TRUE, use TSS instead of the whole gene
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**

```
## download needed data
sesameDataCache(c("Mammal40.address", "genomeInfo.hg38"))

## get all probes overlapping with DNMT3A
probes <- sesameData_getProbesByGene(
  'DNMT3A', "Mammal40", upstream=500, downstream=500)

## get the promoter-associated probes
probes <- sesameData_getProbesByGene('DNMT3A', "Mammal40", promoter = TRUE)
```

---

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,
  chrm_to_exclude = NULL,
  genome = NULL
)
```

**Arguments**

regs	GRanges
chrM	chromosome, when given regs are ignored
beg	begin, 1 if omitted
end	end, chromosome end if omitted
platform	EPICv2, EPIC, HM450, ...
chrM_to_exclude	chromosome to exclude.
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**

```
## download needed data
sesameDataCache(c("Mammal40.address", "genomeInfo.hg38"))

## get probes in a region
library(GenomicRanges)
probes = sesameData_getProbesByRegion(
  GRanges('chr5', IRanges(135313937, 135419936)), platform = 'Mammal40')

## get all probes on chromosome 5
probes = sesameData_getProbesByRegion(chrm = "chr5", platform = "Mammal40")

## get all probes on chromosome X
probes = sesameData_getProbesByRegion(chrm = 'chrX', platform = "Mammal40")

## get all probes on both chromosome X and Y
probes = sesameData_getProbesByRegion(
  chrm = c('chrX', 'chrY'), platform = "Mammal40")

## get all autosomal probes
probes = sesameData_getProbesByRegion(
  chrm_to_exclude = c("chrX", "chrY"), platform = "Mammal40")
```

---

sesameData\_getTxnGRanges

*convert GRangesList to transcript GRanges*

---

**Description**

convert GRangesList to transcript GRanges

**Usage**

```
sesameData_getTxnGRanges(genome = NULL, grl = NULL, merge2gene = FALSE)
```

**Arguments**

genome	hg38, mm10, ...
grl	GRangesList object
merge2gene	merge transcript to genes

**Value**

a GRanges object

**Examples**

```
## all mm10 transcripts
txns <- sesameData_getTxnGRanges("mm10")

## verified protein-coding transcripts
txns[(txns$transcript_type == "protein_coding" & txns$level <= 2)]

## merged to genes
sesameData_getTxnGRanges("mm10", merge2gene = TRUE)
```

---

```
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\_getGenomeInfo, 10

sesameData\_getManifestGRanges, 11

sesameData\_getProbesByGene, 11

sesameData\_getProbesByRegion, 12

sesameData\_getTxnGRanges, 13

sesameData\_txnToGeneGRanges, 14

sesameDataCache, 4

sesameDataCacheAll, 5

sesameDataGet, 5

sesameDataGet\_checkEnv, 6

sesameDataGet\_resetEnv, 6

sesameDataHas, 7

sesameDataList, 7