Package 'sesameData'

April 9, 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. Currently we support human array (HM27, HM450, EPIC), mouse array (MM285) and the HorvathMethylChip40 (Mammal40) array.

Version 1.29.10

License Artistic-2.0

Depends R (>= 4.3.0), 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/issues

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 7.3.3

Encoding UTF-8

git_url https://git.bioconductor.org/packages/sesameData

git_branch devel

git_last_commit 55bde28

git_last_commit_date 2026-02-06

Repository Bioconductor 3.23

Date/Publication 2026-04-09

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

Maintainer Wanding Zhou <zhouwanding@gmail.com>

Contents

`build_GENCODE_gtf` *build GENCODE gtf*

Description

build GENCODE gtf

Usage

```
build_GENCODE_gtf(x)
```

Arguments

x	GENCODE ftp url

Value

GRangesList

`df_master` *Master data frame for all object to cache*

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))

Format

A data frame with 22 columns:

Comments Additional comments

EHID ExperimentHub ID

VERSION sesameData version

IN_USE Logical indicating if the resource is in use

Title Title of the data resource

Description Description of the data resource

BiocVersion Bioconductor version

Genome Genome build (e.g., hg38, mm10)

SourceType Source file type

SourceUrl URL to source

SourceVersion Version of source data

Species Species name

TaxonomyId NCBI Taxonomy ID

Coordinate_1_based Logical indicating if coordinates are 1-based

DataProvider Data provider name

Maintainer Maintainer contact information

RDataClass R data class

DispatchClass Dispatch class for ExperimentHub

RDataPath Path to RData file

Location_Prefix URL prefix for data location

Tags Tags for categorization

Notes Additional notes

Details

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

Value

master sheet of sesameData objects

---

extend *Extend a GRanges*

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 *Cache all SeSAMe data*

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 *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

```
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` *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` *List all SeSAMe data*

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

This function annotates probes based on genomic coordinate overlaps with provided genomic regions (GRanges). 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 http://zwdzwd.github.io/InfiniumAnnotation and provide the following argument ..., genome = sesameAnno_buildManifestGRanges("downloaded_file"),... to this function.
silent	suppress messages

Details

For annotation by probe ID using KYCG databases (rather than by genomic coordinates), see `knowYourCG::annoProbes()`

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", "Mammal40.address"))

## Example 1: Basic usage - annotate with gene names (default)
## When regs=NULL, function defaults to gene annotation
probes <- c("cg14620903", "cg22464003")
anno <- sesameData_annoProbes(probes)
## Returns GRanges with gene_name column

## Example 2: Annotate mouse probes with promoter regions
regs <- sesameData_getTxnGRanges("mm10")
Probe_IDs <- names(sesameData_getManifestGRanges("MM285"))[1:100]
anno <- sesameData_annoProbes(Probe_IDs, promoters(regs), column="gene_name")
## Probes overlapping promoters are annotated with gene names

## Example 3: Get features that overlap with probes
## return_ov_features=TRUE returns the overlapping genomic features
genes <- sesameData_getTxnGRanges("hg38", merge2gene = TRUE)
ov_genes <- sesameData_annoProbes(
    c("cg14620903","cg22464003"), genes, return_ov_features=TRUE)
## Returns GRanges of genes overlapping the probes

## Example 4: Get probes that overlap with features
## return_ov_probes=TRUE returns only overlapping probes
ov_probes <- sesameData_annoProbes(
    c("cg14620903","cg22464003"), genes, return_ov_probes=TRUE)
## Returns GRanges of probes that overlap genes

## Example 5: Control annotation collapsing behavior
## collapse=TRUE (default): multiple annotations concatenated with separator
anno_collapsed <- sesameData_annoProbes(
    c("cg14620903","cg22464003"), genes, column="gene_name",
    collapse=TRUE, sep=";")
## Multiple overlapping genes listed as "GENE1;GENE2;GENE3"

## collapse=FALSE: each probe-feature overlap gets separate entry
anno_expanded <- sesameData_annoProbes(
    c("cg14620903","cg22464003"), genes, column="gene_name",
    collapse=FALSE)
## Probes with multiple overlaps appear multiple times

## Example 6: Choose only first annotation when multiple exist
anno_one <- sesameData_annoProbes(
```

```
    c("cg14620903","cg22464003"), genes, column="gene_name",
    chooseOne=TRUE)
## Each probe gets only the first overlapping gene

## Example 7: Annotate with custom genomic regions
custom_regs <- GRanges(
    seqnames = c("chr5", "chr5"),
    ranges = IRanges(start = c(10000, 135350870),
                     end = c(135350866, 145369531)),
    feature_type = c("enhancer", "silencer"))
anno_custom <- sesameData_annoProbes(
    c("cg14620903","cg22464003"), custom_regs,
    column="feature_type", genome="hg38")

## Note: For annotation by probe ID using KYCG databases
## (rather than genomic coordinates), see knowYourCG::annoProbes()
```

`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, "Mammal40")
```

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 http://zwdzwd.github.io/InfiniumAnnotation 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 http://zwdzwd.github.io/InfiniumAnnotation 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