Package 'sesameData'

October 18, 2022

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 1.14.0

License Artistic-2.0

Depends R (>= 4.1), 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 7.1.2

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

git_branch RELEASE_3_15

git_last_commit 4e4c0ce

git_last_commit_date 2022-04-26

Date/Publication 2022-10-18

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

Maintainer Wanding Zhou <zhouwanding@gmail.com>

R topics documented:

`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))

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

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, use_alternative = FALSE, verbose = FALSE)
```

Arguments

title	title of the data
use_alternative	to use alternative hosting site
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 *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

Annotate Probes by Probe ID

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
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
silent	suppress messages

Value

a GRanges with annotated column

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, "Mammal40")
```

`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`	hg19, hg38, or mm10, inference by default

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, max_distance = 10000)
```

Arguments

Probe_IDs	probe IDs
platform	EPIC, HM450, ... will infer if not given
max_distance	maximum distance to gene (default: 10000)

Value

a GRanges object for overlapping genes

Examples

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

---

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 etc.

Value

GRanges

Examples

```
gr <- sesameData_getManifestGRanges("Mammal40")
```

---

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	hg19, hg38, or mm10, inference by default

Value

GRanges of selected probes

Examples

```
Xprobes <- sesameData_getProbesByChromosome('chrX', "Mammal40")
```

`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,
  platform = NULL,
  upstream = 0,
  downstream = 0,
  genome = NULL
)
```

Arguments

`gene_name`	gene name
`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', "Mammal40", 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

`regs`	GRanges
`chrm`	chromosome, when given regs are ignored
`beg`	begin, 1 if omitted
`end`	end, chromosome end if omitted
`platform`	EPIC, HM450, ...
`genome`	use default if not given

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', "Mammal40")
```

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