

# Package ‘knowYourCG’

November 14, 2025

**Type** Package

**Title** Functional analysis of DNA methylome datasets

**Version** 1.7.0

**Description** KnowYourCG (KYCG) is a supervised learning framework designed for the functional analysis of DNA methylation data. Unlike existing tools that focus on genes or genomic intervals, KnowYourCG directly targets CpG dinucleotides, featuring automated supervised screenings of diverse biological and technical influences, including sequence motifs, transcription factor binding, histone modifications, replication timing, cell-type-specific methylation, and trait-epigenome associations. KnowYourCG addresses the challenges of data sparsity in various methylation datasets, including low-pass Nanopore sequencing, single-cell DNA methylomes, 5-hydroxymethylation profiles, spatial DNA methylation maps, and array-based datasets for epigenome-wide association studies and epigenetic clocks.

**Depends** R (>= 4.4.0)

**URL** <https://github.com/zhou-lab/knowYourCG>

**BugReports** <https://github.com/zhou-lab/knowYourCG/issues>

**License** MIT + file LICENSE

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** false

**RoxygenNote** 7.3.2

**Imports** sesameData, dplyr, methods, rlang, GenomicRanges, IRanges, reshape2, S4Vectors, stats, stringr, utils, ggplot2, ggrepel, tibble, wheatmap, magrittr

**biocViews** Epigenetics, DNAMethylation, Sequencing, SingleCell, Spatial, MethylationArray

**Suggests** testthat (>= 3.0.0), SummarizedExperiment, rmarkdown, knitr, sesame, gprofiler2, ggrastr

**Config/testthat/edition 3****git\_url** <https://git.bioconductor.org/packages/knowYourCG>**git\_branch** devel**git\_last\_commit** 65cf48f**git\_last\_commit\_date** 2025-10-29**Repository** Bioconductor 3.23**Date/Publication** 2025-11-13

**Author** Zhou Wanding [aut] (ORCID: <<https://orcid.org/0000-0001-9126-1932>>),  
 Goldberg David [aut, cre] (ORCID:  
 <<https://orcid.org/0000-0002-9622-4708>>),  
 Fu Hongxiang [ctb]

**Maintainer** Goldberg David <golddc72@penmedicine.upenn.edu>**Contents**

|                                     |    |
|-------------------------------------|----|
| aggregateTestEnrichments . . . . .  | 3  |
| annoProbes . . . . .                | 3  |
| buildGeneDBs . . . . .              | 4  |
| dbStats . . . . .                   | 5  |
| getDBs . . . . .                    | 6  |
| KYCG_plotBar . . . . .              | 7  |
| KYCG_plotDot . . . . .              | 8  |
| KYCG_plotEnrichAll . . . . .        | 9  |
| KYCG_plotLollipop . . . . .         | 10 |
| KYCG_plotManhattan . . . . .        | 10 |
| KYCG_plotMeta . . . . .             | 12 |
| KYCG_plotMetaEnrichment . . . . .   | 12 |
| KYCG_plotPointRange . . . . .       | 13 |
| KYCG_plotSetEnrichment . . . . .    | 14 |
| KYCG_plotVolcano . . . . .          | 14 |
| KYCG_plotWaterfall . . . . .        | 15 |
| linkProbesToProximalGenes . . . . . | 16 |
| listDBGroups . . . . .              | 17 |
| testEnrichment . . . . .            | 17 |
| testEnrichment2 . . . . .           | 19 |
| testEnrichmentFisher . . . . .      | 19 |
| testEnrichmentSEA . . . . .         | 20 |
| testEnrichmentSpearman . . . . .    | 21 |
| testProbeProximity . . . . .        | 21 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>23</b> |
|--------------|-----------|

---

`aggregateTestEnrichments`*Aggregate test enrichment results*

---

**Description**

Aggregate test enrichment results

**Usage**

```
aggregateTestEnrichments(result_list, column = "estimate", return_df = FALSE)
```

**Arguments**

|                          |                                                                |
|--------------------------|----------------------------------------------------------------|
| <code>result_list</code> | a list of results from <code>testEnrichment</code>             |
| <code>column</code>      | the column name to aggregate (Default: <code>estimate</code> ) |
| <code>return_df</code>   | whether to return a merged data frame                          |

**Value**

a matrix for all results

**Examples**

```
## pick some big TFBS-overlapping CpG groups
sesameData::sesameDataCache(data_titles=
c("KYCG.MM285.TFBSconsensus.20220116", "KYCG.MM285.chromHMM.20210210",
"probeIDSignature", "MM285.address"))
cg_lists <- getDBs("MM285.TFBS")
queries <- cg_lists[(sapply(cg_lists, length) > 40000)]
result_list <- lapply(queries, testEnrichment, "MM285.chromHMM")
mtx <- aggregateTestEnrichments(result_list)
```

---

`annoProbes`*Annotate Probe IDs using KYCG databases*

---

**Description**

see `sesameData_annoProbes` if you'd like to annotate by genomic coordinates (in `GRanges`)

Usage

```
annoProbes(  
  probeIDs,  
  databases,  
  db_names = NULL,  
  platform = NULL,  
  sep = ",",  
  indicator = FALSE,  
  silent = FALSE  
)
```

Arguments

|           |                                                                                                             |
|-----------|-------------------------------------------------------------------------------------------------------------|
| probeIDs  | probe IDs in a character vector                                                                             |
| databases | character or actual database (i.e. list of probe IDs)                                                       |
| db_names  | specific database (default to all databases)                                                                |
| platform  | EPIC, MM285 etc. will infer from probe IDs if not given                                                     |
| sep       | delimiter used in paste                                                                                     |
| indicator | return the indicator matrix instead of a concatenated annotation (in the case of have multiple annotations) |
| silent    | suppress message                                                                                            |

Value

named annotation vector, or indicator matrix

Examples

```
sesameData::sesameDataCache(data_titles=  
c("MM285.address","probeIDSignature","KYCG.MM285.designGroup.20210210"))  
probes <- names(sesameData::sesameData_getManifestGRanges("MM285"))  
anno <- annoProbes(probeIDs=probes, "designGroup", silent = TRUE)
```

---

|              |                                              |
|--------------|----------------------------------------------|
| buildGeneDBs | <i>build gene-probe association database</i> |
|--------------|----------------------------------------------|

---

Description

build gene-probe association database

**Usage**

```
buildGeneDBs(
  probeIDs = NULL,
  platform = NULL,
  genome = NULL,
  max_distance = 10000,
  silent = FALSE
)
```

**Arguments**

|              |                                                                   |
|--------------|-------------------------------------------------------------------|
| probeIDs     | the query probe list. If NULL, use all the probes on the platform |
| platform     | HM450, EPIC, MM285, Mammal40, will infer from query if not given  |
| genome       | hg38, mm10, ..., will infer if not given.                         |
| max_distance | probe-gene distance for association                               |
| silent       | suppress messages                                                 |

**Value**

gene databases

**Examples**

```
sesameData::sesameDataCache(data_titles=
c("EPIC.address", "genomeInfo.hg38", "probeIDSignature"))
query <- c("cg04707299", "cg13380562", "cg00480749")
dbs <- buildGeneDBs(query, platform = "EPIC")
testEnrichment(query, dbs, platform = "EPIC")
```

---

|         |                                                                                                    |
|---------|----------------------------------------------------------------------------------------------------|
| dbStats | <i>dbStats aggregates methylation of a given betas matrix over specified database set features</i> |
|---------|----------------------------------------------------------------------------------------------------|

---

**Description**

dbStats aggregates methylation of a given betas matrix over specified database set features

**Usage**

```
dbStats(
  betas,
  databases,
  fun = mean,
  na.rm = TRUE,
  n_min = NULL,
  f_min = 0.1,
  long = FALSE
)
```

**Arguments**

|           |                                                                                           |
|-----------|-------------------------------------------------------------------------------------------|
| betas     | matrix of beta values where probes are on the rows and samples are on the columns         |
| databases | List of vectors corresponding to probe locations for which the features will be extracted |
| fun       | aggregation function, default to mean                                                     |
| na.rm     | whether to remove NA                                                                      |
| n_min     | min number of non-NA for aggregation function to apply, overrides f_min                   |
| f_min     | min fraction of non-NA for aggregation function to apply                                  |
| long      | produce long-form result                                                                  |

**Value**

matrix with samples on the rows and database set on the columns

**Examples**

```
library(SummarizedExperiment)
sesameData::sesameDataCache(data_titles=
c("MM285.467.SE.tissue20Kprobes","KYCG.MM285.probeType.20210630"))
se <- sesameData::sesameDataGet("MM285.467.SE.tissue20Kprobes")
head(dbStats(assay(se), "MM285.probeType")[,1:3])
sesameData::sesameDataGet_resetEnv()
```

---

getDBs

---

*Get databases by full or partial names of the database group(s)*


---

**Description**

Get databases by full or partial names of the database group(s)

**Usage**

```
getDBs(
  group_nms,
  db_names = NULL,
  platform = NULL,
  summary = FALSE,
  allow_multi = FALSE,
  type = NULL,
  silent = FALSE
)
```

**Arguments**

|             |                                                                   |
|-------------|-------------------------------------------------------------------|
| group_nms   | database group names                                              |
| db_names    | name of the database, fetch only the given databases              |
| platform    | EPIC, HM450, MM285, ... If given, will restrict to that platform. |
| summary     | return a summary of database instead of db itself                 |
| allow_multi | allow multiple groups to be returned for                          |
| type        | numerical, categorical, default: all                              |
| silent      | no messages each query.                                           |

**Value**

a list of databases, return NULL if no database is found

**Examples**

```
sesameData::sesameDataCache(data_titles=
c("KYCG.MM285.chromHMM.20210210", "KYCG.MM285.probeType.20210630"))
dbs <- getDBs("MM285.chromHMM")
dbs <- getDBs(c("MM285.chromHMM", "MM285.probeType"))
```

---

KYCG\_plotBar

---

*Bar plot to show most enriched CG groups from testEnrichment*


---

**Description**

The input data frame should have an "estimate" and a "FDR" columns.

**Usage**

```
KYCG_plotBar(df, y = "-log10(FDR)", n = 20, order_by = "FDR", label = FALSE)
```

**Arguments**

|          |                                           |
|----------|-------------------------------------------|
| df       | KYCG result data frame                    |
| y        | the column to be plotted on y-axis        |
| n        | number of CG groups to plot               |
| order_by | the column by which CG groups are ordered |
| label    | whether to label significant bars         |

**Details**

Top CG groups are determined by estimate (descending order).

**Value**

grid plot object

**Examples**

```
KYCG_plotBar(data.frame(
  estimate=runif(10,0,10), FDR=runif(10,0,1), nD=10,
  overlap=as.integer(runif(10,0,30)), group="g", dbname=seq_len(10)))
```

---

|              |                                                                     |
|--------------|---------------------------------------------------------------------|
| KYCG_plotDot | <i>Dot plot to show most enriched CG groups from testEnrichment</i> |
|--------------|---------------------------------------------------------------------|

---

**Description**

The input data frame should have an "estimate" and a "FDR" columns.

**Usage**

```
KYCG_plotDot(
  df,
  y = "-log10(FDR)",
  n = 20,
  order_by = "FDR",
  title = "Enriched Knowledgebases",
  label_by = "dbname",
  size_by = "overlap",
  color_by = "estimate",
  short_label = FALSE
)
```

**Arguments**

|             |                                           |
|-------------|-------------------------------------------|
| df          | KYCG result data frame                    |
| y           | the column to be plotted on y-axis        |
| n           | number of CG groups to plot               |
| order_by    | the column by which CG groups are ordered |
| title       | plot title                                |
| label_by    | the column for label                      |
| size_by     | the column by which CG group size plot    |
| color_by    | the column by which CG groups are colored |
| short_label | omit group in label                       |

**Details**

Top CG groups are determined by estimate (descending order).

**Value**

grid plot object (by ggplot)

**Examples**

```
KYCG_plotDot(data.frame(
  estimate=runif(10,0,10), FDR=runif(10,0,1), nD=runif(10,10,20),
  overlap=as.integer(runif(10,0,30)), group="g", dbname=seq_len(10)))
```

---

|                    |                                    |
|--------------------|------------------------------------|
| KYCG_plotEnrichAll | <i>plot enrichment test result</i> |
|--------------------|------------------------------------|

---

**Description**

plot enrichment test result

**Usage**

```
KYCG_plotEnrichAll(
  df,
  fdr_max = 25,
  n_label = 15,
  min_estimate = 0,
  short_label = TRUE
)
```

**Arguments**

|              |                                   |
|--------------|-----------------------------------|
| df           | test enrichment result data frame |
| fdr_max      | maximum fdr for capping           |
| n_label      | number of database to label       |
| min_estimate | minimum estimate                  |
| short_label  | use short label                   |

**Value**

grid object

**Examples**

```
query <- getDBs("MM285.designGroup")["PGCMeth"]
res <- testEnrichment(query, platform="MM285")
KYCG_plotEnrichAll(res)
```

---

|                   |                                                                                  |
|-------------------|----------------------------------------------------------------------------------|
| KYCG_plotLollipop | <i>creates a lollipop plot of log(estimate) given data with fields estimate.</i> |
|-------------------|----------------------------------------------------------------------------------|

---

**Description**

creates a lollipop plot of log(estimate) given data with fields estimate.

**Usage**

```
KYCG_plotLollipop(df, label_column = "dbname", n = 20)
```

**Arguments**

- df                      DataFrame where each row is a database name with its estimate.
- label\_column          column in df to be used as the label (default: dbname)
- n                       Integer representing the number of top enrichments to report. Optional. (Default: 10)

**Value**

ggplot lollipop plot

**Examples**

```
KYCG_plotLollipop(data.frame(  
  estimate=runif(10,0,10), FDR=runif(10,0,1), nD=runif(10,10,20),  
  overlap=as.integer(runif(10,0,30)), group="g",  
  dbname=as.character(seq_len(10))))
```

---

|                    |                                                                            |
|--------------------|----------------------------------------------------------------------------|
| KYCG_plotManhattan | <i>KYCG_plotManhattan makes a manhattan plot to summarize EWAS results</i> |
|--------------------|----------------------------------------------------------------------------|

---

**Description**

KYCG\_plotManhattan makes a manhattan plot to summarize EWAS results

**Usage**

```
KYCG_plotManhattan(
  vals,
  platform = NULL,
  genome = NULL,
  title = NULL,
  rasterize = FALSE,
  rasterize_thres = 3,
  label_min = 100,
  col = c("wheat1", "sienna3"),
  ylabel = "Value"
)
```

**Arguments**

|                              |                                                                                                                                                                                                                                                                                                                                  |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>vals</code>            | named vector of values (P,Q etc), vector name is Probe ID.                                                                                                                                                                                                                                                                       |
| <code>platform</code>        | String corresponding to the type of platform to use for retrieving GRanges coordinates of probes. Either MM285, EPIC, HM450, or HM27. If it is not provided, it will be inferred from the query set probeIDs (Default: NA).                                                                                                      |
| <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 ..., genome = sesameAnno_buildManifestGRanges("downloaded_file"),... to this function. |
| <code>title</code>           | title for plot                                                                                                                                                                                                                                                                                                                   |
| <code>rasterize</code>       | if true use ggrastr to rasterize non-significant data.                                                                                                                                                                                                                                                                           |
| <code>rasterize_thres</code> | the threshold of rasterize                                                                                                                                                                                                                                                                                                       |
| <code>label_min</code>       | Threshold above which data points will be labelled with Probe ID                                                                                                                                                                                                                                                                 |
| <code>col</code>             | color                                                                                                                                                                                                                                                                                                                            |
| <code>ylabel</code>          | y-axis label                                                                                                                                                                                                                                                                                                                     |

**Value**

a ggplot object

**Examples**

```
## see vignette for examples
```

---

`KYCG_plotMeta`*Plot meta gene or other meta genomic features*

---

**Description**

Plot meta gene or other meta genomic features

**Usage**

```
KYCG_plotMeta(betas, platform = NULL)
```

**Arguments**

`betas` a named numeric vector or a matrix (row: probes; column: samples)  
`platform` if not given and x is a SigDF, will be inferred the meta features

**Value**

a grid plot object

**Examples**

```
library(sesameData)
library(sesame)
sdf <- sesameDataGet("EPIC.1.SigDF")
KYCG_plotMeta(getBetas(sdf))
```

---

`KYCG_plotMetaEnrichment`*Plot meta gene or other meta genomic features*

---

**Description**

Plot meta gene or other meta genomic features

**Usage**

```
KYCG_plotMetaEnrichment(result_list)
```

**Arguments**

`result_list` one or a list of testEnrichment

**Value**

a grid plot object

**Examples**

```
cg_lists <- getDBs("MM285.TFBS")
queries <- cg_lists[(sapply(cg_lists, length) > 40000)]
result_list <- lapply(queries, testEnrichment,
  "MM285.metagene", silent=TRUE, platform="MM285")

KYCG_plotMetaEnrichment(result_list)
```

---

|                     |                                                                                                    |
|---------------------|----------------------------------------------------------------------------------------------------|
| KYCG_plotPointRange | <i>Plot point range for a list of enrichment testing results against the same set of databases</i> |
|---------------------|----------------------------------------------------------------------------------------------------|

---

**Description**

Plot point range for a list of enrichment testing results against the same set of databases

**Usage**

```
KYCG_plotPointRange(result_list)
```

**Arguments**

result\_list      a list of testEnrichment resultsx

**Value**

grid plot object

**Examples**

```
## pick some big TFBS-overlapping CpG groups
cg_lists <- getDBs("MM285.TFBS")
queries <- cg_lists[(sapply(cg_lists, length) > 40000)]
result_list <- lapply(queries, testEnrichment,
  "MM285.chromHMM", platform="MM285")
KYCG_plotPointRange(result_list)
```

---

KYCG\_plotSetEnrichment

*Plot Set Enrichment*


---

### Description

Plot Set Enrichment

### Usage

```
KYCG_plotSetEnrichment(result, n_sample = 1000, n_presence = 200)
```

### Arguments

|            |                                                                                                |
|------------|------------------------------------------------------------------------------------------------|
| result     | result object as returned from an element of the list of testEnrichmentSEA(..., prepPlot=TRUE) |
| n_sample   | number of CpGs to sample                                                                       |
| n_presence | number of overlap to sample for the plot                                                       |

### Value

grid object for plot

### Examples

```
query <- getDBs("KYCG.MM285.designGroup")[[ "VMR" ]]  
db <- getDBs("MM285.seqContextN", "distToTSS")  
res <- testEnrichmentSEA(query, db, prepPlot = TRUE)  
KYCG_plotSetEnrichment(res[[1]])
```

---

|                  |                                                                                                                                             |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| KYCG_plotVolcano | <i>creates a volcano plot of <math>-\log_2(p.value)</math> and <math>\log(estimate)</math> given data with fields estimate and p.value.</i> |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------|

---

### Description

creates a volcano plot of  $-\log_2(p.value)$  and  $\log(estimate)$  given data with fields estimate and p.value.

### Usage

```
KYCG_plotVolcano(df, label_by = "dbname", alpha = 0.05)
```

**Arguments**

|          |                                                                                             |
|----------|---------------------------------------------------------------------------------------------|
| df       | DataFrame where each field is a database name with two fields for the estimate and p.value. |
| label_by | column in df to be used as the label (default: dbname)                                      |
| alpha    | Float representing the cut-off alpha value for the plot. Optional. (Default: 0.05)          |

**Value**

ggplot volcano plot

**Examples**

```
KYCG_plotVolcano(data.frame(
  estimate=runif(10,0,10), FDR=runif(10,0,1), nD=runif(10,10,20),
  overlap=as.integer(runif(10,0,30)), group="g", dbname=seq_len(10)))
```

---

|                    |                                                                       |
|--------------------|-----------------------------------------------------------------------|
| KYCG_plotWaterfall | <i>create a waterfall plot of log(estimate) given test enrichment</i> |
|--------------------|-----------------------------------------------------------------------|

---

**Description**

create a waterfall plot of log(estimate) given test enrichment

**Usage**

```
KYCG_plotWaterfall(
  df,
  order_by = "Log2(OR)",
  size_by = "-log10(FDR)",
  label_by = "dbname",
  n_label = 10
)
```

**Arguments**

|          |                                                                     |
|----------|---------------------------------------------------------------------|
| df       | data frame where each row is a database with test enrichment result |
| order_by | the column by which CG groups are ordered                           |
| size_by  | the column by which CG group size plot                              |
| label_by | column in df to be used as the label (default: dbname)              |
| n_label  | number of datapoints to label                                       |

**Value**

grid

**Examples**

```
library(SummarizedExperiment)
library(sesameData)
df <- rowData(sesameDataGet('MM285.tissueSignature'))
query <- df$Probe_ID[df$branch == "fetal_brain" & df$type == "Hypo"]
results <- testEnrichment(query, "TFBS", platform="MM285")
KYCG_plotWaterfall(results)
```

---

linkProbesToProximalGenes

*find genes in genomic proximity to given Infinium probes*

---

**Description**

This is a convenient function that uses `sesameData_getGenomeInfo()` to retrieve stored gene models.

**Usage**

```
linkProbesToProximalGenes(probeIDs, platform = NULL, genome = NULL)
```

**Arguments**

|          |                                                                             |
|----------|-----------------------------------------------------------------------------|
| probeIDs | character vector of probe IDs                                               |
| platform | HM450, EPIC, EPICv2, MM285, MSA, ..., will infer from probe ID if not given |
| genome   | hg38, hg19, mm10, this is usually inferred from platform.                   |

**Details**

For finer control, such as taking only genes by their promoters, please use `sesameData_getTxnGRanges` followed by `sesameData_annoProbes()`. See code of this convenient function for details.

**Value**

a data frame annotate gene list linked to each given probes

**Examples**

```
library(SummarizedExperiment)
probes = rowData(sesameData::sesameDataGet('MM285.tissueSignature'))$Probe_ID[1:10]
linkProbesToProximalGenes(probes, platform = "MM285")
```

---

|              |                                  |
|--------------|----------------------------------|
| listDBGroups | <i>List database group names</i> |
|--------------|----------------------------------|

---

**Description**

List database group names

**Usage**

```
listDBGroups(filter = NULL, path = NULL, type = NULL)
```

**Arguments**

|        |                                            |
|--------|--------------------------------------------|
| filter | keywords for filtering                     |
| path   | file path to downloaded knowledgebase sets |
| type   | categorical, numerical (default: all)      |

**Value**

a list of db group names

**Examples**

```
head(listDBGroups("chromHMM"))  
## or listDBGroups(path = "~/Downloads")
```

---

|                |                                                                               |
|----------------|-------------------------------------------------------------------------------|
| testEnrichment | <i>testEnrichment tests for the enrichment of query in knowledgebase sets</i> |
|----------------|-------------------------------------------------------------------------------|

---

**Description**

testEnrichment tests for the enrichment of query in knowledgebase sets

**Usage**

```
testEnrichment(  
  query,  
  databases = NULL,  
  universe = NULL,  
  alternative = "greater",  
  include_genes = FALSE,  
  platform = NULL,  
  silent = FALSE,  
  mtc_by_group = TRUE,  
  mtc_method = "fdr"  
)
```

**Arguments**

|               |                                                                                                                                                                                           |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| query         | For array input, it is a vector of probes of interest (e.g., significant differential methylated probes). For sequencing data input, it expect the file name for YAME-compressed CG sets. |
| databases     | List of vectors corresponding to the database sets of interest with associated meta data as an attribute to each element. Optional. (Default: NA)                                         |
| universe      | Vector of probes in the universe set containing all of the probes to be considered in the test. If it is not provided, it will be inferred from the provided platform. (Default: NA).     |
| alternative   | "two.sided", "greater", or "less"                                                                                                                                                         |
| include_genes | include gene link enrichment testing                                                                                                                                                      |
| platform      | String corresponding to the type of platform to use. Either MM285, EPIC, HM450, or HM27. If it is not provided, it will be inferred from the query set probeIDs (Default: NA).            |
| silent        | output message? (Default: FALSE)                                                                                                                                                          |
| mtc_by_group  | perform multiple testing correction within knowledgebase groups (Default: TRUE)                                                                                                           |
| mtc_method    | method for multiple test correction (default: fdr)                                                                                                                                        |

**Value**

A data frame containing features corresponding to the test estimate, p-value, and type of test.

**Examples**

```
library(SummarizedExperiment)
sesameData::sesameDataCache(data_titles=
c("MM285.tissueSignature", "KYCG.MM285.chromHMM.20210210", "MM285.address"))
df <- rowData(sesameData::sesameDataGet("MM285.tissueSignature"))
probes <- df$Probe_ID[df$branch == "B_cell"]
res <- testEnrichment(probes, "chromHMM", platform="MM285")
sesameData::sesameDataGet_resetEnv()

## Not run:
# Define temporary directory and file URLs
temp_dir <- tempdir()
knowledgebase <- file.path(temp_dir, "ChromHMM.20220414.cm")
query <- file.path(temp_dir, "single_cell_10_samples.cg")
# URLs for the knowledgebase and query files
knowledgebase_url <- "https://github.com/zhou-lab/KYCGKB_mm10/raw/refs/heads/main/ChromHMM.20220414.cm"
query_url <- "https://github.com/zhou-lab/YAME/raw/refs/heads/main/test/input/single_cell_10_samples.cg"
# Download the files
download.file(knowledgebase_url, destfile = knowledgebase)
download.file(query_url, destfile = query)
# Confirm file download
list.files(temp_dir)
res = testEnrichment(query, knowledgebase)

## End(Not run)
```

---

|                 |                                                     |
|-----------------|-----------------------------------------------------|
| testEnrichment2 | <i>Test enrichment from YAME-compressed CG sets</i> |
|-----------------|-----------------------------------------------------|

---

**Description**

Test enrichment from YAME-compressed CG sets

**Usage**

```
testEnrichment2(  
  query_fn,  
  knowledge_fn,  
  universe_fn = NULL,  
  alternative = "greater"  
)
```

**Arguments**

|              |                                |
|--------------|--------------------------------|
| query_fn     | File path to query             |
| knowledge_fn | File path to knowledgebase     |
| universe_fn  | optional file path to universe |
| alternative  | greater, less                  |

**Value**

A single concatenated string.

**Examples**

```
if (.Platform$OS.type!="windows") {  
  kfn = system.file("extdata", "chromhmm.cm", package = "knowYourCG")  
  qfn = system.file("extdata", "onecell.cg", package = "knowYourCG")  
  testEnrichment2(qfn, kfn)  
}
```

---

|                      |                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------|
| testEnrichmentFisher | <i>testEnrichmentFisher uses Fisher's exact test to estimate the association between two categorical variables.</i> |
|----------------------|---------------------------------------------------------------------------------------------------------------------|

---

**Description**

Estimates log2 Odds ratio

**Usage**

```
testEnrichmentFisher(query, database, universe, alternative = "greater")
```

**Arguments**

|             |                                                                                                                  |
|-------------|------------------------------------------------------------------------------------------------------------------|
| query       | Vector of probes of interest (e.g., significant probes)                                                          |
| database    | Vectors corresponding to the database set of interest with associated meta data as an attribute to each element. |
| universe    | Vector of probes in the universe set containing all of                                                           |
| alternative | greater or two.sided (default: greater) the probes to be considered in the test. (Default: NULL)                 |

**Value**

A DataFrame with the estimate/statistic, p-value, and name of test for the given results.

---

|                   |                                                                                                                     |
|-------------------|---------------------------------------------------------------------------------------------------------------------|
| testEnrichmentSEA | <i>uses the GSEA-like test to estimate the association of a categorical variable against a continuous variable.</i> |
|-------------------|---------------------------------------------------------------------------------------------------------------------|

---

**Description**

estimate represent enrichment score and negative estimate indicate a test for depletion

**Usage**

```
testEnrichmentSEA(
  query,
  databases,
  platform = NULL,
  silent = FALSE,
  precise = FALSE,
  prepPlot = FALSE
)
```

**Arguments**

|           |                                                                                            |
|-----------|--------------------------------------------------------------------------------------------|
| query     | query, if numerical, expect categorical database, if categorical expect numerical database |
| databases | database, numerical or categorical, but needs to be different from query                   |
| platform  | EPIC, MM285, ..., infer if not given                                                       |
| silent    | suppress message (default: FALSE)                                                          |
| precise   | whether to compute precise p-value (up to numerical limit) of interest.                    |
| prepPlot  | return the raw enrichment scores and presence vectors for plotting                         |

**Value**

A DataFrame with the estimate/statistic, p-value, and name of test for the given results.

**Examples**

```
sesameData::sesameDataCache(data_titles=
c("KYCG.MM285.designGroup.20210210", "KYCG.MM285.seqContextN.20210630",
"probeIDSignature"))
query <- getDBs("KYCG.MM285.designGroup")[[ "TSS" ]]
res <- testEnrichmentSEA(query, "MM285.seqContextN")
```

---

testEnrichmentSpearman

*testEnrichmentSpearman uses the Spearman statistical test to estimate the association between two continuous variables.*

---

**Description**

testEnrichmentSpearman uses the Spearman statistical test to estimate the association between two continuous variables.

**Usage**

```
testEnrichmentSpearman(num_query, num_db)
```

**Arguments**

|           |                                                                                                                          |
|-----------|--------------------------------------------------------------------------------------------------------------------------|
| num_query | named numeric vector of probes of interest where names are probe IDs (e.g significant probes)                            |
| num_db    | List of vectors corresponding to the database set of interest with associated meta data as an attribute to each element. |

**Value**

A DataFrame with the estimate/statistic, p-value, and name of test for the given results.

---

|                    |                                                                                                                      |
|--------------------|----------------------------------------------------------------------------------------------------------------------|
| testProbeProximity | <i>testProbeProximity tests if a query set of probes share closer genomic proximity than if randomly distributed</i> |
|--------------------|----------------------------------------------------------------------------------------------------------------------|

---

**Description**

testProbeProximity tests if a query set of probes share closer genomic proximity than if randomly distributed

**Usage**

```
testProbeProximity(
  probeIDs,
  gr = NULL,
  platform = NULL,
  iterations = 100,
  bin_size = 1500
)
```

**Arguments**

|            |                                                                                                                                                                                |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| probeIDs   | Vector of probes of interest (e.g., significant probes)                                                                                                                        |
| gr         | GRanges to draw samples and compute genomic distances                                                                                                                          |
| platform   | String corresponding to the type of platform to use. Either MM285, EPIC, HM450, or HM27. If it is not provided, it will be inferred from the query set probeIDs (Default: NA). |
| iterations | Number of random samples to generate null distribution (Default: 100).                                                                                                         |
| bin_size   | the poisson interval size for computing neighboring hits                                                                                                                       |

**Value**

list containing a dataframe for the poisson statistics and a data frame for the probes in close proximity

**Examples**

```
sesameData::sesameDataCache(data_titles=
  c("MM285.tissueSignature", "MM285.address", "probeIDSignature"))
library(SummarizedExperiment)
df <- rowData(sesameData::sesameDataGet("MM285.tissueSignature"))
probes <- df$Probe_ID[df$branch == "B_cell"]
res <- testProbeProximity(probeIDs=probes, platform="MM285")
sesameData::sesameDataGet_resetEnv()
```

# Index

[aggregateTestEnrichments](#), [3](#)  
[annoProbes](#), [3](#)  
  
[buildGeneDBs](#), [4](#)  
  
[dbStats](#), [5](#)  
  
[getDBs](#), [6](#)  
  
[KYCG\\_plotBar](#), [7](#)  
[KYCG\\_plotDot](#), [8](#)  
[KYCG\\_plotEnrichAll](#), [9](#)  
[KYCG\\_plotLollipop](#), [10](#)  
[KYCG\\_plotManhattan](#), [10](#)  
[KYCG\\_plotMeta](#), [12](#)  
[KYCG\\_plotMetaEnrichment](#), [12](#)  
[KYCG\\_plotPointRange](#), [13](#)  
[KYCG\\_plotSetEnrichment](#), [14](#)  
[KYCG\\_plotVolcano](#), [14](#)  
[KYCG\\_plotWaterfall](#), [15](#)  
  
[linkProbesToProximalGenes](#), [16](#)  
[listDBGroups](#), [17](#)  
  
[testEnrichment](#), [17](#)  
[testEnrichment2](#), [19](#)  
[testEnrichmentFisher](#), [19](#)  
[testEnrichmentSEA](#), [20](#)  
[testEnrichmentSpearman](#), [21](#)  
[testProbeProximity](#), [21](#)