

# Package ‘knowYourCG’

December 19, 2024

**Type** Package

**Title** Functional analysis of DNA methylome datasets

**Version** 1.3.1

**Description** knowYourCG automates the functional analysis of DNA methylation data. The package tests the enrichment of discrete CpG probes across thousands of curated biological and technical features. GSEA-like analysis can be performed on continuous methylation data query sets. knowYourCG can also take beta matrices as input to perform feature aggregation over the curated database sets.

**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** 7.3.1

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

**biocViews** Epigenetics, DNAMethylation, MethylationArray

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

**Config/testthat/edition** 3

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

**git\_branch** devel

**git\_last\_commit** 5970122

**git\_last\_commit\_date** 2024-12-16

**Repository** Bioconductor 3.21

**Date/Publication** 2024-12-18

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

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

## Contents

aggregateTestEnrichments . . . . .	2
annoProbes . . . . .	3
buildGeneDBs . . . . .	4
dbStats . . . . .	5
getDBs . . . . .	6
KYCG_plotBar . . . . .	7
KYCG_plotDot . . . . .	7
KYCG_plotEnrichAll . . . . .	9
KYCG_plotLollipop . . . . .	9
KYCG_plotManhattan . . . . .	10
KYCG_plotMeta . . . . .	11
KYCG_plotMetaEnrichment . . . . .	12
KYCG_plotPointRange . . . . .	12
KYCG_plotSetEnrichment . . . . .	13
KYCG_plotVolcano . . . . .	14
KYCG_plotWaterfall . . . . .	14
listDBGroups . . . . .	15
testEnrichment . . . . .	16
testEnrichmentFisher . . . . .	17
testEnrichmentSEA . . . . .	17
testEnrichmentSpearman . . . . .	18
testGO . . . . .	19
testProbeProximity . . . . .	19
<b>Index</b>	<b>21</b>

---

aggregateTestEnrichments

*Aggregate test enrichment results*

---

### Description

Aggregate test enrichment results

### Usage

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

**Arguments**

result\_list      a list of results from testEnrichment  
 column            the column name to aggregate (Default: estimate)  
 return\_df        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)
```

**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

**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**

<code>group_nms</code>	database group names
<code>db_names</code>	name of the database, fetch only the given databases
<code>platform</code>	EPIC, HM450, MM285, ... If given, will restrict to that platform.
<code>summary</code>	return a summary of database instead of db itself
<code>allow_multi</code>	allow multiple groups to be returned for
<code>type</code>	numerical, categorical, default: all
<code>silent</code>	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**

<code>df</code>	KYCG result data frame
<code>y</code>	the column to be plotted on y-axis
<code>n</code>	number of CG groups to plot
<code>order_by</code>	the column by which CG groups are ordered
<code>label</code>	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`*Dot plot to show most enriched CG groups from testEnrichment*

---

**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 Databases",  
  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 *plot enrichment test result*

---

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

---

**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,
  label_min = 100,
  col = c("wheat1", "sienna3"),
  ylabel = "Value"
)
```

**Arguments**

vals	named vector of values (P,Q etc), vector name is Probe ID.
platform	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).

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.
title	title for plot
label_min	Threshold above which data points will be labelled with Probe ID
col	color
ylabel	y-axis label

**Value**

a ggplot object

**Examples**

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

---

KYCG_plotMeta	<i>Plot meta gene or other meta genomic features</i>
---------------	--

---

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

---

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

---

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

---

**Description**

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

**Usage**

```
KYCG_plotWaterfall(
  df,
  order_by = "Log2(OR)",
  size_by = " $-\log_{10}(FDR)$ ",
  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)
```

---

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 a set of probes (query set) in a number of features (database sets).</i>
----------------	--

---

### Description

testEnrichment tests for the enrichment of a set of probes (query set) in a number of features (database sets).

### Usage

```
testEnrichment(
  probeIDs,
  databases = NULL,
  universe = NULL,
  alternative = "greater",
  include_genes = FALSE,
  platform = NULL,
  silent = FALSE
)
```

### Arguments

probeIDs	Vector of probes of interest (e.g., significant probes)
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)

### 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()

```

---

testEnrichmentFisher *testEnrichmentFisher uses Fisher's exact test to estimate the association between two categorical variables.*

---

### 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 *uses the GSEA-like test to estimate the association of a categorical variable against a continuous variable.*

---

### 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.

---

testGO	<i>tests Gene Ontology of genes overlapping CpG query</i>
--------	---

---

**Description**

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

**Usage**

```
testGO(probeIDs, platform = NULL, organism = "hsapiens", gene_name = TRUE, ...)
```

**Arguments**

probeIDs	Vector of CpG probes IDs or a data frame with gene_name column, usually the output of testEnrichment() function
platform	EPIC, MM285, ..., infer if not given
organism	The organism corresponding to the CpG platform or genes in gene_name column
gene_name	If query is data frame from testEnrichment output, whether to use the gene_name column. If set to FALSE, TFBS will be used (default: FALSE)
...	Additional arguments to sesameData_getGenesByProbes and gost()

**Value**

A list of enriched terms and meta data from gprofiler2 output

**Examples**

```
library(SummarizedExperiment)
sesameData::sesameDataCache(data_titles=
c("MM285.tissueSignature", "probeIDSignature",
"MM285.address", "genomeInfo.mm10"))
df <- rowData(sesameData::sesameDataGet('MM285.tissueSignature'))
query <- df$Probe_ID[df$branch == "fetal_liver" & df$type == "Hypo"]
res <- testGO(query, platform="MM285")
```

---

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](#), 2  
[annoProbes](#), 3

[buildGeneDBs](#), 4

[dbStats](#), 5

[getDBs](#), 6

[KYCG\\_plotBar](#), 7

[KYCG\\_plotDot](#), 7

[KYCG\\_plotEnrichAll](#), 9

[KYCG\\_plotLollipop](#), 9

[KYCG\\_plotManhattan](#), 10

[KYCG\\_plotMeta](#), 11

[KYCG\\_plotMetaEnrichment](#), 12

[KYCG\\_plotPointRange](#), 12

[KYCG\\_plotSetEnrichment](#), 13

[KYCG\\_plotVolcano](#), 14

[KYCG\\_plotWaterfall](#), 14

[listDBGroups](#), 15

[testEnrichment](#), 16

[testEnrichmentFisher](#), 17

[testEnrichmentSEA](#), 17

[testEnrichmentSpearman](#), 18

[testGO](#), 19

[testProbeProximity](#), 19