Package 'rGREAT'

April 10, 2023

Type Package

Title GREAT Analysis - Functional Enrichment on Genomic Regions

Version 2.0.2

Date 2022-11-21

Depends R (>= 3.6.0), GenomicRanges, IRanges, methods

Imports graphics, rjson, GetoptLong (>= 0.0.9), RCurl, utils, stats,
GlobalOptions, shiny, DT, GenomicFeatures, digest, GO.db,
progress, circlize, AnnotationDbi,
TxDb.Hsapiens.UCSC.hg19.knownGene,
TxDb.Hsapiens.UCSC.hg38.knownGene, org.Hs.eg.db, RColorBrewer,
S4Vectors, GenomeInfoDb, foreach, doParallel, Rcpp

Suggests testthat (>= 0.3), knitr, rmarkdown, BiocManager,
org.Mm.eg.db, msigdbr, KEGGREST, reactome.db

Enhances BioMartGOGeneSets, UniProtKeywords

VignetteBuilder knitr

biocViews GeneSetEnrichment, GO, Pathways, Software, Sequencing,
WholeGenome, GenomeAnnotation, Coverage

Description GREAT (Genomic Regions Enrichment of Annotations Tool) is a type of
functional enrichment analysis directly performed on genomic regions. This package
implements the GREAT algorithm (the local GREAT analysis), also it supports directly
interacting with the GREAT web service (the online GREAT analysis). Both analysis
can be viewed by a Shiny application. rGREAT by default supports more than 600 organisms
and a large number of gene set collections, as well as self-provided gene sets and
organisms from users. Additionally, it implements a general method for dealing
with background regions.

URL https://github.com/jokergoo/rGREAT,
http://great.stanford.edu/public/html/

License MIT + file LICENSE

LinkingTo Rcpp

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

git_branch RELEASE_3_16

git_last_commit a8049fc

git_last_commit_date 2022-11-21

Date/Publication 2023-04-10

Author Zuguang Gu [aut, cre] (<<https://orcid.org/0000-0002-7395-8709>>)

Maintainer Zuguang Gu <z.gu@dkfz.de>

R **topics documented:**

Index **39**

availableCategories-GreatJob-method

Available ontology categories of the GREAT job

Description

Available ontology categories of the GREAT job

Usage

```
## S4 method for signature 'GreatJob'
availableCategories(object)
```

Arguments

object   A [GreatJob-class](#) object returned by [submitGreatJob](#).

Details

The values of the supported categories sometime change. You should run the function to get the real-time values. The meaning of categories returned is quite self-explained by the name.

Value

The returned value is a vector of categories.

Author(s)

Zuguang gu <z.gu@dkfz.de>

Examples

```
job = readRDS(system.file("extdata", "GreatJob.rds", package = "rGREAT"))
availableCategories(job)
```

---

availableOntologies-GreatJob-method

All available ontology names of the GREAT job

Description

All available ontology names of the GREAT job

Usage

```
## S4 method for signature 'GreatJob'
availableOntologies(object, category = NULL)
```

Arguments

object   A GreatJob-class object returned by submitGreatJob.

category   one or multiple categories. All available categories can be got by availableCategories.

Details

The values of the supported ontologies sometime change. You should run the function to get the real-time values. The meaning of ontology returned is quite self-explained by the name.

Value

The returned values is a vector of ontologies.

Author(s)

Zuguang gu <z.gu@dkfz.de>

Examples

```
job = readRDS(system.file("extdata", "GreatJob.rds", package = "rGREAT"))
availableOntologies(job)
```

---

extendTSS   *Extend TSS*

Description

Extend TSS

Usage

```
extendTSS(gene, seqlengths = NULL, genome = NULL,
    gene_id_type = NULL, mode = "basalPlusExt", basal_upstream = 5000,
    basal_downstream = 1000, extension = 1000000,
    verbose = great_opt$verbose, .attr = list())
```

Arguments

gene	A [GRanges](#) object of gene (or TSS) coordinates.
seqlengths	A named vector of chromosome lengths. If it is not provided, it is taken by seqlengths(gene).
genome	UCSC genome can be set here, then seqlengths will be automatically retrieved from UCSC server.
gene_id_type	Gene ID types in gene. You need to set this argument if you use built-in gene sets in [great](#) so that genes can be correctly mapped. The value can only be one of "SYMBOL", "ENTREZ", "ENSEMBL" and "REFSEQ".
mode	The mode to extend TSS. Value should be one of 'basalPlusExt', 'twoClosest' and 'oneClosest'. See "Details" section.
basal_upstream	In 'basalPlusExt' mode, number of base pairs extending to the upstream of TSS to form the basal domains.
basal_downstream	
	In 'basalPlusExt' mode, number of base pairs extending to the downstream of TSS to form the basal domains.
extension	Extensions from the basal domains.
verbose	Whether to print messages.
.attr	Only used internally.

Details

Following are general explanations of the three modes for extending TSS:

basalPlusExt 1. TSS are extended into basal domains (e.g. by upstream 5kb, downstream 1kb); 2. basal domains are sorted by their genomic coordinates; 3. each basal domain is extended to its both sides until it reaches the next TSS's basal domain or it reaches the maximal extension (e.g. 1000kb).

twoClosest 1. TSS are sorted by their genomic coordinates; 2. each TSS is extended to its both sides until it reaches the next TSS or it reaches the maximal extension (e.g. 1000kb).

oneClosest 1. TSS are sorted by their genomic coordinates; 2. each TSS is extended to its both sides until it reaches the middle point of itself and the next TSS or it reaches the maximal extension (e.g. 1000kb).

The official explanation is at `https://great-help.atlassian.net/wiki/spaces/GREAT/pages/655443/Association+Rules` .

Value

A [GRanges](#) object with one meta column 'gene_id'.

Examples

```
# There is no example
NULL
```

---

extendTSSFromDataFrame

Extend TSS

Description

Extend TSS

Usage

```
extendTSSFromDataFrame(df, seqlengths, genome = NULL,
    strand = NULL, gene_id = NULL,
    gene_id_type = NULL, verbose = great_opt$verbose, ...)
```

Arguments

df	A bed-like data frame where the first three columns should be chromosomes, start positions, end positions. It does not matter whether regions correspond to genes or TSS.
seqlengths	A named vector of chromosome lengths.
genome	UCSC genome can be set here, then `seqlengths` will be automatically retrieved from UCSC server.
strand	The strand information can be provided in `df` as a column named "strand" or as a column with "+"/"-"/"*", or the strand information can be provided as a vector and be assigned to this argument.
gene_id	The gene ID information can be provided in `df` as a column named "gene_id", or it can be provided as a vector and be assigned to this argument.
gene_id_type	Gene ID types in `df`. You need to set this argument if you use built-in gene sets in [great](#) so that genes can be correctly mapped. The value can only be one of "SYMBOL", "ENTREZ", "ENSEMBL" and "REFSEQ".
verbose	Whether to print messages.
...	All pass to [extendTSS](#).

Value

A [GRanges](#) object with one meta column 'gene_id'.

Examples

```
# There is no example
NULL
```

---

extendTSSFromOrgDb   *Extend TSS*

Description

Extend TSS

Usage

```
extendTSSFromOrgDb(orgdb, verbose = great_opt$verbose, ...)
```

Arguments

orgdb	Name of "org.*" packages from Bioconductor. All supported OrgDb packages are in rGREAT:::BIOC_ANNO_PKGS$orgdb.
verbose	Whether to print messages.
...	All pass to [extendTSS](#).

Value

A [GRanges](#) object with one meta column 'gene_id'.

Examples

```
if(FALSE) {
extendTSSFromOrgDb("Org.Hs.eg.db")
extendTSSFromOrgDb("hg19")
}
```

---

extendTSSFromTxDb   *Extend TSS*

Description

Extend TSS

Usage

```
extendTSSFromTxDb(txdb, verbose = great_opt$verbose, ...)
```

Arguments

txdb	Name of "TxDb.*" packages from Bioconductor. All supported TxDb packages are in rGREAT:::BIOC_ANNO_PKGS$txdb. Note short genome version can also be used here such as "hg19" or "hg19.knownGene".
verbose	Whether to print messages.
...	All pass to extendTSS.

Value

A GRanges object with one meta column 'gene_id'.

Examples

```
if(FALSE) {
extendTSSFromTxDb("TxDb.Hsapiens.UCSC.hg19.knownGene")
extendTSSFromTxDb("hg19")
}
```

---

getEnrichmentTable-dispatch
                          Method dispatch page for getEnrichmentTable

---

Description

Method dispatch page for getEnrichmentTable.

Dispatch

getEnrichmentTable can be dispatched on following classes:

- getEnrichmentTable,GreatObject-method, GreatObject-class class method
- getEnrichmentTable,GreatJob-method, GreatJob-class class method

Examples

```
# no example
NULL
```

getEnrichmentTable-GreatJob-method

Get a single enrichment table from GREAT web server

Description

Get a single enrichment table from GREAT web server

Usage

```
## S4 method for signature 'GreatJob'
getEnrichmentTable(object, ontology, ...)
```

Arguments

object	A GreatJob-class object returned by submitGreatJob.
ontology	A single ontology names. Valid values are in availableOntologies.
...	All pass to getEnrichmentTables,GreatJob-method.

Value

A data frame of the enrichment results for a single ontology.

Examples

```
job = readRDS(system.file("extdata", "GreatJob.rds", package = "rGREAT"))
tb = getEnrichmentTable(job, ontology = "GO Molecular Function")
head(tb)
```

getEnrichmentTable-GreatObject-method
Get enrichment table

Description

Get enrichment table

Usage

```
## S4 method for signature 'GreatObject'
getEnrichmentTable(object, min_region_hits = 5)
```

Arguments

object   A [GreatObject-class](#) object returned by [great](#).

min_region_hits

                  Minimal number of input regions overlapping to the geneset associated regions.

Details

Note: adjusted p-values are re-calculated based on min_region_hits.

Value

A data frame of enrichment results

Examples

```
obj = readRDS(system.file("extdata", "GreatObject.rds", package = "rGREAT"))
getEnrichmentTable(obj)
```

getEnrichmentTables-dispatch
Method dispatch page for getEnrichmentTables

Description

Method dispatch page for getEnrichmentTables.

Dispatch

getEnrichmentTables can be dispatched on following classes:

- [getEnrichmentTables,GreatObject-method](#), [GreatObject-class](#) class method
- [getEnrichmentTables,GreatJob-method](#), [GreatJob-class](#) class method

Examples

```
# no example
NULL
```

```
getEnrichmentTables-GreatJob-method
```
Get enrichment tables from GREAT web server

Description

Get enrichment tables from GREAT web server

Usage

```
## S4 method for signature 'GreatJob'
getEnrichmentTables(object, ontology = NULL, category = "GO",
    request_interval = 10, max_tries = 100, download_by = c("json", "tsv"),
    verbose = TRUE)
```

Arguments

object	A [GreatJob-class](#) object returned by [submitGreatJob](#).
ontology	Ontology names. Valid values are in [availableOntologies](#). ontology is prior to category argument.
category	Pre-defined ontology categories. One category can contain more than one ontologies. Valid values are in [availableCategories](#)
request_interval	
	Time interval for two requests. Default is 300 seconds.
max_tries	Maximal times for automatically reconnecting GREAT web server.
download_by	Internally used.
verbose	Whether to print messages.

Value

The structure of the data frames are same as the tables available on GREAT website.

See

[availableOntologies](#), [availableCategories](#)

Author(s)

Zuguang gu <z.gu@dkfz.de>

Examples

```
job = readRDS(system.file("extdata", "GreatJob.rds", package = "rGREAT"))
tbl = getEnrichmentTables(job)
names(tbl)
head(tbl[[1]])
job

tbl = getEnrichmentTables(job, ontology = "GO Molecular Function")
tbl = getEnrichmentTables(job, category = "GO")
```

getEnrichmentTables-GreatObject-method
Get enrichment table

Description

Get enrichment table

Usage

```
## S4 method for signature 'GreatObject'
getEnrichmentTables(object, ...)
```

Arguments

object   A GreatObject-class object returned by great.

...   All passed to getEnrichmentTable,GreatObject-method.

Details

Please use getEnrichmentTable,GreatObject-method directly.

Value

A data frame of enrichment results

Examples

```
# There is no example
NULL
```

---

getGapFromUCSC *Get gap regions from UCSC*

Description

Get gap regions from UCSC

Usage

```
getGapFromUCSC(genome, seqnames = NULL)
```

Arguments

genome   UCSC genome, such as "hg19".

seqnames   A vector of chromosome names.

Value

A [GRanges](#) object.

Examples

```
getGapFromUCSC("hg19")
```

---

getGeneSetsFromBioMart

Get gene sets from BioMart

Description

Get gene sets from BioMart

Usage

```
getGeneSetsFromBioMart(dataset, ontology = "bp")
```

Arguments

dataset   Name of the dataset.

ontology   Value should be bp, mf or cc.

Details

GO gene sets are from `BioMartGOGeneSets::getBioMartGOGeneSets`.

Value

A list of vectors where each vector contains Ensembl IDs annotated to a GO term.

Examples

```
# There is no example
NULL
```

---

getGenesFromGencode   *Get Gencode genes*

Description

Get Gencode genes

Usage

```
getGenesFromGencode(version)
```

Arguments

version   Gencode version, e.g. v19 for human, vM21 for mouse.

Details

Only the protein coding genes.

Value

A [GRanges](#) object.

Examples

```
# There is no example
NULL
```

---

getGREATDefaultTSS   *Get built-in TSS from GREAT*

Description

Get built-in TSS from GREAT

Usage

```
getGREATDefaultTSS(genome)
```

Arguments

genome   Only support "hg19", "hg38", "mm10", "mm9". Files are downloaded from
                https://great-help.atlassian.net/wiki/spaces/GREAT/pages/655445/
                Genes .

Value

A [GRanges](#) object.

Examples

```
# There is no example
NULL
```

---

getRefSeqGenesFromUCSC

                          Get RefSeq genes from UCSC

Description

Get RefSeq genes from UCSC

Usage

```
getRefSeqGenesFromUCSC(genome, subset = c("RefSeqSelect", "RefSeqCurated"))
```

Arguments

genome   UCSC genome, such as "hg19".

subset   Subset of RefSeq genes. See https://genome.ucsc.edu/cgi-bin/hgTrackUi?
                db=hg38&g=refSeqComposite .

Value

A [GenomicRanges](#) object.

Examples

```
# There is no example
NULL
```

getRegionGeneAssociations-dispatch
                        Method dispatch page for getRegionGeneAssociations

Description

Method dispatch page for getRegionGeneAssociations.

Dispatch

getRegionGeneAssociations can be dispatched on following classes:

- [getRegionGeneAssociations,GreatObject-method](#), [GreatObject-class](#) class method
- [getRegionGeneAssociations,GreatJob-method](#), [GreatJob-class](#) class method

Examples

```
# no example
NULL
```

getRegionGeneAssociations-GreatJob-method
                        Get region-gene associations

Description

Get region-gene associations

Usage

```
## S4 method for signature 'GreatJob'
getRegionGeneAssociations(object, ontology = NULL, term_id = NULL,
    request_interval = 10, max_tries = 100, verbose = great_opt$verbose)
```

Arguments

`object`	A [`GreatJob-class`](#) object returned by [`submitGreatJob`](#).
`ontology`	ontology name
`term_id`	Term id in the selected ontology.
`request_interval`	
	Time interval for two requests. Default is 300 seconds.
`max_tries`	Maximal times for automatically reconnecting GREAT web server.
`verbose`	Whether to show messages.

Value

A [`GRanges`](#) object. Please the two meta columns are in formats of `CharacterList` and `IntegerList` because a region may associate to multiple genes.

Please note, the distance is from the middle points of input regions to TSS.

Author(s)

Zuguang gu <z.gu@dkfz.de>

Examples

```
job = readRDS(system.file("extdata", "GreatJob.rds", package = "rGREAT"))

gr = getRegionGeneAssociations(job)
gr
```

---

getRegionGeneAssociations-GreatObject-method
Get region-gene associations

Description

Get region-gene associations

Usage

```
## S4 method for signature 'GreatObject'
getRegionGeneAssociations(object, term_id = NULL, by_middle_points = FALSE,
    use_symbols = TRUE)
```

Arguments

`object`	A [`GreatObject-class`](#) object returned by [`great`](#).
`term_id`	Term ID.
`by_middle_points`	
	Whether the distances are calculated from the middle points of input regions?
`use_symbols`	Whether to use gene symbols

Value

A [GRanges](#) object. Please the two meta columns are in formats of `CharacterList` and `IntegerList` because a region may associate to multiple genes.

Examples

```
obj = readRDS(system.file("extdata", "GreatObject.rds", package = "rGREAT"))
getRegionGeneAssociations(obj)
```

---

getTSS   *Get the internally used TSS*

Description

Get the internally used TSS

Usage

```
getTSS(tss_source, biomart_dataset = NULL)
```

Arguments

tss_source   The same format as in [great](#).

biomart_dataset
                  The same format as in [great](#).

Value

A [GRanges](#) object.

Examples

```
# There is no example
NULL
```

---

great   *Perform GREAT analysis*

Description

Perform GREAT analysis

Usage

```
great(gr, gene_sets, tss_source, biomart_dataset = NULL,
    min_gene_set_size = 5, mode = "basalPlusExt", basal_upstream = 5000,
    basal_downstream = 1000, extension = 1000000,
    extended_tss = NULL, background = NULL, exclude = "gap",
    cores = 1, verbose = great_opt$verbose)
```

Arguments

gr   A [GRanges](#) object. This is the input regions. It is important to keep consistent
                        for the chromosome names of the input regions and the internal TSS regions.
                        Use [getTSS](#) to see the format of internal TSS regions.

gene_sets   A single string of defaulty supported gene sets collections (see the full list in
                        "Genesets" section), or a named list of vectors where each vector correspond to
                        a gene set.

tss_source   Source of TSS. See "TSS" section.

biomart_dataset

                        The value should be in BioMartGOGeneSets::supportedOrganisms.

min_gene_set_size

                        Minimal size of gene sets.

mode   The mode to extend genes. Value should be one of 'basalPlusExt', 'twoClosest'
                        and 'oneClosest'. See [extendTSS](#) for details.

basal_upstream   In 'basalPlusExt' mode, number of base pairs extending to the upstream of TSS
                        to form the basal domains.

basal_downstream

                        In 'basalPlusExt' mode, number of base pairs extending to the downstream of
                        TSS to form the basal domains.

extension   Extensions from the basal domains.

extended_tss   If your organism is not defaulty supported, you can first prepare one by [extendTSSFromDataFrame](#)
                        or [extendTSS](#), and set the object to this argument. Please see more examples in
                        the vignette.

background   Background regions. The value can also be a vector of chromosome names.

exclude   Regions that are excluded from analysis such as gap regions (which can be get
                        by [getGapFromUCSC](#)). The value can also be a vector of chromosome names. It
                        also allows a special character value "gap" so that gap regions for corresponding
                        organism will be removed from the analysis.

cores   Number of cores to use.

verbose   Whether to print messages.

Details

When `background` or `exclude` is set, the analysis is restricted in the background regions, still by using Binomial method. Note this is different from the original GREAT method which uses Fisher's exact test if background regions is set. See submitGreatJob for explanations.

By default, gap regions are excluded from the analysis.

Value

A GreatObject-class object. The following methods can be applied on it:

- getEnrichmentTable,GreatObject-method to retrieve the result table.
- getRegionGeneAssociations,GreatObject-method to get the associations between input regions and genes.
- plotRegionGeneAssociations,GreatObject-method to plot the associations bewteen input regions and genes.
- shinyReport,GreatObject-method to view the results by a shiny application.

Tss

rGREAT supports TSS from many organisms. The value of tss_source should be encoded in a special format:

- Name of `TxDb.*` packages. Supported packages are in `rGREAT:::BIOC_ANNO_PKGS$txdb`.
- Genome version of the organism, e.g. "hg19". Then the corresponding TxDb will be used.
- In a format of `RefSeqCurated:$genome` where $genome is the genome version of an organism, such as hg19. RefSeqCurated subset will be used.
- In a format of `RefSeqSelect:$genome` where $genome is the genome version of an organism, such as hg19. RefSeqSelect subset will be used.
- In a format of `Gencode_v$version` where $version is gencode version, such as 19 (for human) or M21 for mouse. Gencode protein coding genes will be used.
- In a format of `GREAT:$genome`, where $genome can only be mm9, mm10, hg19, hg38. The TSS from GREAT will be used.

Genesets

rGREAT supports the following built-in GO gene sets for all organisms (note "GO:" can be omitted):

"GO:BP": Biological Process, from GO.db package.

"GO:CC": Cellular Component, from GO.db package.

"GO:MP": Molecular Function, from GO.db pacakge.

rGREAT also supports built-in gene sets collections from MSigDB (note this is only for human, "msigdb:" can be omitted):

"msigdb:H" Hallmark gene sets.

"msigdb:C1" Positional gene sets.

"msigdb:C2" Curated gene sets.

"msigdb:C2:CGP" C2 subcategory: chemical and genetic perturbations gene sets.

"msigdb:C2:CP" C2 subcategory: canonical pathways gene sets.

"msigdb:C2:CP:BIOCARTA" C2 subcategory: BioCarta subset of CP.

"msigdb:C2:CP:KEGG" C2 subcategory: KEGG subset of CP.

"msigdb:C2:CP:PID" C2 subcategory: PID subset of CP.

"msigdb:C2:CP:REACTOME" C2 subcategory: REACTOME subset of CP.

"msigdb:C2:CP:WIKIPATHWAYS" C2 subcategory: WIKIPATHWAYS subset of CP.

"msigdb:C3" Regulatory target gene sets.

"msigdb:C3:MIR:MIRDB" miRDB of microRNA targets gene sets.

"msigdb:C3:MIR:MIR_LEGACY" MIR_Legacy of MIRDB.

"msigdb:C3:TFT:GTRD" GTRD transcription factor targets gene sets.

"msigdb:C3:TFT:TFT_LEGACY" TFT_Legacy.

"msigdb:C4" Computational gene sets.

"msigdb:C4:CGN" C4 subcategory: cancer gene neighborhoods gene sets.

"msigdb:C4:CM" C4 subcategory: cancer modules gene sets.

"msigdb:C5" Ontology gene sets.

"msigdb:C5:GO:BP" C5 subcategory: BP subset.

"msigdb:C5:GO:CC" C5 subcategory: CC subset.

"msigdb:C5:GO:MF" C5 subcategory: MF subset.

"msigdb:C5:HPO" C5 subcategory: human phenotype ontology gene sets.

"msigdb:C6" Oncogenic signature gene sets.

"msigdb:C7" Immunologic signature gene sets.

"msigdb:C7:IMMUNESIGDB" ImmuneSigDB subset of C7.

"msigdb:C7:VAX" C7 subcategory: vaccine response gene sets.

"msigdb:C8" Cell type signature gene sets.

If the defaultly supported TxDb is used, Entrez gene ID is always used as the main gene ID. If you provide a self-defined `gene_sets` or `extended_tss`, you need to make sure they two have the same gene ID types.

Biomart

rGREAT supports a large number of organisms of which the information is retrieved from Ensembl BioMart. The name of a BioMart dataset can be assigned to argument `biomart_dataset`. All supported organisms can be found with `BioMartGOGeneSets::supportedOrganisms`.

Examples

```
if(FALSE) {
gr = randomRegions(genome = "hg19")
res = great(gr, "MSigDB:H", "txdb:hg19")
res = great(gr, "MSigDB:H", "TxDb.Hsapiens.UCSC.hg19.knownGene")
res = great(gr, "MSigDB:H", "RefSeq:hg19")
res = great(gr, "MSigDB:H", "GREAT:hg19")
res = great(gr, "MSigDB:H", "Gencode_v19")
res = great(gr, "GO:BP", "hsapiens_gene_ensembl")
}
```

GreatJob *Constructor method for GreatJob class*

Description

Constructor method for GreatJob class

Usage

```
GreatJob(...)
```

Arguments

`...`	arguments.

Details

There is no public constructor method for the GreatJob-class.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

GreatJob-class *Class to store and retrieve GREAT results*

Description

Class to store and retrieve GREAT results

Details

After submitting request to GREAT server, the generated results will be available on GREAT server for some time. The GreatJob-class is defined to store parameters that user has set and result tables what were retrieved from GREAT server.

Constructor

Users don't need to construct by hand, submitGreatJob is used to generate a GreatJob-class instance.

Workflow

After submitting request to GREAT server, users can perform following steps:

- getEnrichmentTables,GreatJob-method to get enrichment tables for selected ontologies catalogues.
- plotRegionGeneAssociations,GreatJob-method to plot associations between regions and genes
- getRegionGeneAssociations,GreatJob-method to get a GRanges object which contains associations bewteen regions and genes.
- shinyReport,GreatJob-method to view the results by a shiny application.

Author(s)

Zuguang gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

GreatObject *Constructor method for GreatObject class*

Description

Constructor method for GreatObject class

Usage

```
GreatObject(...)
```

Arguments

... arguments.

Details

There are following methods that can be applied on `GreatObject-class` object:

- `getEnrichmentTable,GreatObject-method` to retrieve the result table.
- `getRegionGeneAssociations,GreatObject-method` to get the associations between input regions and genes.
- `plotRegionGeneAssociations,GreatObject-method` to plot the associations bewteen input regions and genes.
- `shinyReport,GreatObject-method` to view the results by a shiny application.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
NULL
```

GreatObject-class *Class for local GREAT analysis*

Description

Class for local GREAT analysis

Details

great returns A GreatObject-class object.

Examples

```
# There is no example
NULL
```

great_opt *Global parameters for rGREAT*

Description

Global parameters for rGREAT

Usage

```
great_opt(..., RESET = FALSE, READ.ONLY = NULL, LOCAL = FALSE, ADD = FALSE)
```

Arguments

...	Arguments for the parameters, see "details" section
RESET	Reset to default values.
READ.ONLY	Please ignore.
LOCAL	Pllease ignore.
ADD	Please ignore.

Details

There are following parameters:

verbose Whether to show messages.

Examples

```
great_opt
```

plotRegionGeneAssociationGraphs-GreatJob-method
Plot region-gene associations

Description

Plot region-gene associations

Usage

```
## S4 method for signature 'GreatJob'
plotRegionGeneAssociationGraphs(object, ...)
```

Arguments

object	A [GreatJob-class](#) object returned by [submitGreatJob](#).
...	All passed to [plotRegionGeneAssociations,GreatJob-method](#).

Details

This function will be removed in the future, please use [plotRegionGeneAssociations,GreatJob-method](#) instead.

Examples

```
# There is no example
NULL
```

plotRegionGeneAssociations-dispatch
Method dispatch page for plotRegionGeneAssociations

Description

Method dispatch page for plotRegionGeneAssociations.

Dispatch

plotRegionGeneAssociations can be dispatched on following classes:

- [plotRegionGeneAssociations,GreatJob-method](#), [GreatJob-class](#) class method
- [plotRegionGeneAssociations,GreatObject-method](#), [GreatObject-class](#) class method

Examples

```
# no example
NULL
```

---

plotRegionGeneAssociations-GreatJob-method

Plot region-gene associations

Description

Plot region-gene associations

Usage

```
## S4 method for signature 'GreatJob'
plotRegionGeneAssociations(object, ontology = NULL, term_id = NULL, which_plot = 1:3,
    request_interval = 10, max_tries = 100, verbose = great_opt$verbose)
```

Arguments

object	A [GreatJob-class](#) object returned by [submitGreatJob](#).
ontology	A single ontology names. Valid values are in [availableOntologies](#).
term_id	Term id in the selected ontology
which_plot	Which plots to draw? The value should be in 1, 2, 3. See "Details" section for explanation.
request_interval	Time interval for two requests. Default is 300 seconds.
max_tries	Maximal times for automatically reconnecting GREAT web server.
verbose	Whether to show messages.

Details

There are following figures:

- Association between regions and genes (which_plot = 1).
- Distribution of distance to TSS (which_plot = 2).
- Distribution of absolute distance to TSS (which_plot = 3).

If ontology and term_id are set, only regions and genes corresponding to selected ontology term will be used. Valid value for ontology is in [availableOntologies](#) and valid value for term_id is from 'id' column in the table which is returned by [getEnrichmentTables](#).

Author(s)

Zuguang gu <z.gu@dkfz.de>

Examples

```
job = readRDS(system.file("extdata", "GreatJob.rds", package = "rGREAT"))

plotRegionGeneAssociations(job)
plotRegionGeneAssociations(job, which_plot = 1)
# Do not use other term_id for this example, or you need to generate a new `job` object.
plotRegionGeneAssociations(job, ontology = "GO Molecular Function",
    term_id = "GO:0004984")
```

---

plotRegionGeneAssociations-GreatObject-method
Plot region-gene associations

Description

Plot region-gene associations

Usage

```
## S4 method for signature 'GreatObject'
plotRegionGeneAssociations(object, term_id = NULL, which_plot = 1:3)
```

Arguments

object	A [GreatObject-class](#) object returned by [great](#).
term_id	Term ID.
which_plot	Which plots to draw? The value should be in 1, 2, 3. See "Details" section for explanation.

Details

There are following figures:

- Association between regions and genes (which_plot = 1).
- Distribution of distance to TSS (which_plot = 2).
- Distribution of absolute distance to TSS (which_plot = 3).

Examples

```
obj = readRDS(system.file("extdata", "GreatObject.rds", package = "rGREAT"))
plotRegionGeneAssociations(obj)
```

plotVolcano-dispatch   *Method dispatch page for plotVolcano*

Description

Method dispatch page for plotVolcano.

Dispatch

plotVolcano can be dispatched on following classes:

- [plotVolcano,GreatObject-method](), [GreatObject-class]() class method
- [plotVolcano,GreatJob-method](), [GreatJob-class]() class method

Examples

```
# no example
NULL
```

plotVolcano-GreatJob-method
                         Make volcano plot

Description

Make volcano plot

Usage

```
## S4 method for signature 'GreatJob'
plotVolcano(object, ontology, min_region_hits = 5,
    x_values = c("fold_enrichment", "z-score"),
    y_values = c("p_value", "p_adjust"),
    main = NULL)
```

Arguments

object	A [GreatJob-class]() object returned by [submitGreatJob]().
ontology	A single ontology names. Valid values are in [availableOntologies]().
min_region_hits	
	Minimal number of input regions overlapping to the geneset associated regions.
x_values	Which values for the x-axis.
y_values	Which values for the y-axis.
main	Title of the plot.

Details

Since the enrichment is an over-representation test, it is only the half volcano.

Examples

```
# There is no example
NULL
```

```
plotVolcano-GreatObject-method
                              Make volcano plot
```

Description

Make volcano plot

Usage

```
## S4 method for signature 'GreatObject'
plotVolcano(object, min_region_hits = 5,
    x_values = c("fold_enrichment", "z-score"),
    y_values = c("p_value", "p_adjust"),
    main = NULL)
```

Arguments

object	A [GreatObject-class](#) object returned by [great](#).
min_region_hits	
	Minimal number of input regions overlapping to the geneset associated regions.
x_values	Which values for the x-axis.
y_values	Which values for the y-axis.
main	Title of the plot.

Details

Since the enrichment is an over-representation test, it is only the half volcano.

Examples

```
# There is no example
NULL
```

---

randomRegions   *Generate random regions*

Description

Generate random regions

Usage

```
randomRegions(genome = NULL, nr = 1000, seqlengths = NULL,
    width_fun = function(n) runif(n, min = 1000, max = 10000))
```

Arguments

genome	UCSC genome version, e.g. "hg19".
nr	Number of regions.
seqlengths	Alternatively, you can also specify a named vector of seqlengths (chromosome lengths).
width_fun	A function which defines the distribution of region widths.

Details

The number of regions per chromosome is proportional to the chromsome length.

Examples

```
gr = randomRegions(genome = "hg19")
quantile(width(gr))
```

---

randomRegionsFromBioMartGenome

Generate random regions from a BioMart genome

Description

Generate random regions from a BioMart genome

Usage

```
randomRegionsFromBioMartGenome(biomart_dataset, nr = 1000, ...)
```

Arguments

biomart_dataset	A BioMart dataset. Values should be in BioMartGOGeneSets::supportedOrganisms.
nr	Number of regions.
...	Pass to randomRegions.

Details

The number of regions per chromosome is proportional to the chromsome length.

Examples

```
if(FALSE) {
    # Giant panda
    gr = randomRegionsFromBioMartGenome("amelanoleuca_gene_ensembl")
}
```

---

read_gmt   *Read gmt gene sets file*

Description

Read gmt gene sets file

Usage

```
read_gmt(x, from = NULL, to = NULL, orgdb = NULL)
```

Arguments

x	The file name of a .gmt file.
from	Gene ID type in the original gmt file. Value can only take values in 'ENTREZ/SYMBOL/ENSEMBL/REFSEQ'.
to	Gene ID type that you want to convert to. Value can only take values in 'ENTREZ/SYMBOL/ENSEMBL/REFSEQ'.
orgdb	The name of an OrgDb database.

Value

A named list of vectors.

Examples

```
read_gmt(url("http://dsigdb.tanlab.org/Downloads/D2_LINCS.gmt"))
```

reduce_by_start_and_end

Reduce by start and end

Description

Reduce by start and end

Usage

```
reduce_by_start_and_end(s, e)
```

Arguments

s	Start positions. Sorted.
e	End positions. Sorted.

Details

Only internally used.

Value

Sum of total widths of the reduced regions.

Examples

```
# There is no example
NULL
```

---

shinyReport-dispatch   *Method dispatch page for shinyReport*

Description

Method dispatch page for shinyReport.

Dispatch

shinyReport can be dispatched on following classes:

- shinyReport,GreatObject-method, GreatObject-class class method
- shinyReport,GreatJob-method, GreatJob-class class method

Examples

```
# no example
NULL
```

---

shinyReport-GreatJob-method

Shiny app on the GreatJob object

Description

Shiny app on the GreatJob object

Usage

```
## S4 method for signature 'GreatJob'
shinyReport(object)
```

Arguments

object   The `GreatJob` object returned by `submitGreatJob`.

Value

A shiny app object.

Examples

```
if(FALSE) {
# pseudo code
job = submitGreatJob(...)
shinyReport(job)
}
```

---

shinyReport-GreatObject-method

Shiny app on the GreatObject object

Description

Shiny app on the GreatObject object

Usage

```
## S4 method for signature 'GreatObject'
shinyReport(object)
```

Arguments

object   The `GreatObject` object returned by [great](#).

Value

A shiny app object.

Examples

```
if(FALSE) {
# pseudo code
obj = great(...)
shinyReport(obj)
}
```

---

submitGreatJob   *Perform online GREAT analysis*

Description

Perform online GREAT analysis

Usage

```
submitGreatJob(gr, bg = NULL,
    gr_is_zero_based    = FALSE,
    species             = "hg19",
    includeCuratedRegDoms = TRUE,
    rule                = c("basalPlusExt", "twoClosest", "oneClosest"),
    adv_upstream        = 5.0,
    adv_downstream      = 1.0,
    adv_span            = 1000.0,
    adv_twoDistance     = 1000.0,
    adv_oneDistance     = 1000.0,
    request_interval = 60,
    max_tries = 10,
    version = DEFAULT_VERSION,
    base_url = "http://great.stanford.edu/public/cgi-bin",
    help = TRUE)
```

Arguments

gr   A [GRanges](#) object or a data frame which contains at least three columns (chr, start and end).

bg   Not supported any more. See explanations in section "When_background_regions_are_set".

gr_is_zero_based

       Are start positions in `gr` zero-based?

species   Species. "hg38", "hg19", "mm10", "mm9" are supported in GREAT version
                   4.x.x, "hg19", "mm10", "mm9", "danRer7" are supported in GREAT version
                   3.x.x and "hg19", "hg18", "mm9", "danRer7" are supported in GREAT version
                   2.x.x.

includeCuratedRegDoms

                   Whether to include curated regulatory domains, see [https://great-help.atlassian.](https://great-help.atlassian.net/wiki/spaces/GREAT/pages/655443/Association+Rules#AssociationRules-CuratedRegulat)
                   [net/wiki/spaces/GREAT/pages/655443/Association+Rules#AssociationRules-CuratedRegulat](https://great-help.atlassian.net/wiki/spaces/GREAT/pages/655443/Association+Rules#AssociationRules-CuratedRegulat)
                   .

rule   How to associate genomic regions to genes. See 'Details' section.

adv_upstream   Unit: kb, only used when rule is basalPlusExt.

adv_downstream   Unit: kb, only used when rule is basalPlusExt.

adv_span   Unit: kb, only used when rule is basalPlusExt.

adv_twoDistance

                   Unit: kb, only used when rule is twoClosest.

adv_oneDistance

                   Unit: kb, only used when rule is oneClosest.

request_interval

                   Time interval for two requests. Default is 300 seconds.

max_tries   Maximal times for aotumatically reconnecting GREAT web server.

version   Version of GREAT. The value should be "4.0.4", "3.0.0", "2.0.2". Shorten ver-
                   sion numbers can also be used, such as using "4" or "4.0" is same as "4.0.4".

base_url   the url of cgi-bin path, only used when it is explicitly specified.

help   Whether to print help messages.

Details

Note: On Aug 19 2019 GREAT released version 4([https://great-help.atlassian.net/wiki/](https://great-help.atlassian.net/wiki/spaces/GREAT/pages/655442/Version+History)
[spaces/GREAT/pages/655442/Version+History](https://great-help.atlassian.net/wiki/spaces/GREAT/pages/655442/Version+History)) where it supports hg38 genome and removes
some ontologies such pathways. submitGreatJob still takes hg19 as default. hg38 can be spec-
ified by the species = "hg38" argument. To use the older versions such as 3.0.0, specify as
submitGreatJob(..., version = "3.0.0").

Note it does not use the standard GREAT API. This function directly send data to GREAT web
server by HTTP POST.

Following text is copied from GREAT web site (http://great.stanford.edu/public/html/)

Explanation of rule and settings with names started with 'adv_' (advanced settings):

basalPlusExt Mode 'Basal plus extension'. Gene regulatory domain definition: Each gene is as-
signed a basal regulatory domain of a minimum distance upstream and downstream of the TSS
(regardless of other nearby genes, controlled by adv_upstream and adv_downstream argu-
ment). The gene regulatory domain is extended in both directions to the nearest gene's basal
domain but no more than the maximum extension in one direction (controlled by adv_span).

twoClosest Mode 'Two nearest genes'. Gene regulatory domain definition: Each gene is assigned
a regulatory domain that extends in both directions to the nearest gene's TSS (controlled by
adv_twoDistance) but no more than the maximum extension in one direction.

oneClosest Mode 'Single nearest gene'. Gene regulatory domain definition: Each gene is assigned a regulatory domain that extends in both directions to the midpoint between the gene's TSS and the nearest gene's TSS (controlled by adv_oneDistance) but no more than the maximum extension in one direction.

Value

A `GreatJob-class` object which can be used to get results from GREAT server. The following methods can be applied on it:

- `getEnrichmentTables,GreatObject-method` to retreive the result tables.
- `getRegionGeneAssociations,GreatObject-method` to get the associations between input regions and genes.
- `plotRegionGeneAssociations,GreatObject-method` to plot the associations bewteen input regions and genes.
- `shinyReport,GreatObject-method` to view the results by a shiny application.

When_background_regions_are_set

Note when bg argument is set to a list of background regions, GREAT uses a completely different test!

When bg is set, gr should be exactly subset of bg. For example, let's say a background region list contains five regions: [1, 10], [15, 23], [34, 38], [40, 49], [54, 63], gr can only be a subset of the five regions, which means gr can take [15, 23], [40, 49], but it cannot take [16, 20], [39, 51]. In this setting, regions are taken as single units and Fisher's exact test is applied for calculating the enrichment (by testing number of regions in the 2x2 contigency table).

Check `https://great-help.atlassian.net/wiki/spaces/GREAT/pages/655452/File+Formats#FileFormats-Whatshouldmybackgroundregionsfilecontain?` for more explanations.

Please note from rGREAT 1.99.0, setting bg is not supported any more and this argument will be removed in the future. You can either directly use GREAT website or use other Bioconductor packages such as "LOLA" to perform the Fisher's exact test-based analysis.

If you want to restrict the input regions to background regions (by intersections) and still to apply Binomial test there, please consider to use local GREAT by `great`.

Author(s)

Zuguang gu <z.gu@dkfz.de>

See Also

`great` for the local implementation of GREAT algorithm.

Examples

```
set.seed(123)
gr = randomRegions(nr = 1000, genome = "hg19")
job = submitGreatJob(gr)
job
```

```
# more parameters can be set for the job
if(FALSE) { # suppress running it when building the package
    # current GREAT version is 4.0.4
    job = submitGreatJob(gr, genome = "hg19")
    job = submitGreatJob(gr, adv_upstream = 10, adv_downstream = 2, adv_span = 2000)
    job = submitGreatJob(gr, rule = "twoClosest", adv_twoDistance = 2000)
    job = submitGreatJob(gr, rule = "oneClosest", adv_oneDistance = 2000)
}
```

Index