Package: clusterProfiler (via r-universe)

November 24, 2026

Type Package

Title A Universal Enrichment Tool for Interpreting Omics Data

Version 4.22.0

Maintainer Guangchuang Yu <guangchuangyu@gmail.com>

Description A universal tool for interpreting functional characteristics of omics data. It supports Over-Representation Analysis (ORA) and Gene Set Enrichment Analysis (GSEA) for both coding and non-coding genomics data of thousands of species. It provides a unified and tidy interface to access, manipulate, and visualize enrichment results. A key capability is the simultaneous analysis and comparison of datasets from multiple treatments or time points. It also provides high-level access to topology-aware enrichment workflows powered by 'enrichit', including single-network and multi-layer network enrichment for Gene Ontology and KEGG analyses. Furthermore, it integrates Large Language Model (LLM) capabilities to provide automated and insightful interpretation of enrichment results.

Depends R (>= 4.2.0)

Imports aisdk, AnnotationDbi, dplyr, enrichit (>= 0.1.6), enrichplot (>= 1.9.3), ggplot2, GO.db, GOSemSim (>= 2.27.2), gson (>= 0.0.7), httr, igraph, jsonlite, magrittr, methods, plyr, qvalue, rlang, parallel, stats, tidyr, utils, yulab.utils (>= 0.2.3)

Suggests AnnotationHub, ape, BiocManager, digest, DOSE, ggtangle, ggtree, mechgraph, org.Hs.eg.db, quarto, readr, testthat, treeio, withr

Enhances aisdk.mcp, gpsadb

Remotes YuLab-SMU/aisdk.mcp, YuLab-SMU/gpsadb

VignetteBuilder quarto

ByteCompile true

License Artistic-2.0

URL https://yulab-smu.top/contribution-knowledge-mining/

BugReports https://github.com/YuLab-SMU/clusterProfiler/issues

biocViews Annotation, Clustering, GeneSetEnrichment, GO, KEGG, MultipleComparison, Pathways, Reactome, Visualization

Encoding UTF-8

Config/roxygen2/version 8.1.1

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:30:44 UTC

RemoteUrl https://github.com/bioc/clusterProfiler

RemoteRef RELEASE_3_24

RemoteSha dafc6965020219c5214d90620f5dc3c3303ecffd

Contents

add_go_level . . . 4
adjudicate . . . 5
append_kegg_category . . . 7
ask . . . 7
attribute_genes . . . 8
bitr . . . 9
bitr_kegg . . . 10
browseKEGG . . . 11
chat . . . 11
collapse_findings . . . 12
compareCluster . . . 13
consensus_findings . . . 15
DataSet . . . 16
download_KEGG . . . 16
dropGO . . . 17
enrichDAVID . . . 17
enricher . . . 18
enrichGO . . . 19
enrichKEGG . . . 21
enrichMKEGG . . . 22
enrichPC . . . 23
enrichWP . . . 24
evidence . . . 25
evidence_biotrigger . . . 28
evidence_perturbation . . . 29
evidence_ppi . . . 31
evidence_status . . . 32
fork_dialogue . . . 33
get_ppi_network . . . 33
get_wp_organisms . . . 34

getPPI . . . 35
getTaxID . . . 36
getTaxInfo . . . 37
Gff2GeneTable . . . 37
go2ont . . . 38
go2term . . . 38
gofilter . . . 39
groupGO . . . 39
groupGOResult-class . . . 41
GSEA . . . 41
gseGO . . . 43
gseKEGG . . . 44
gseMKEGG . . . 46
gsePC . . . 47
gseWP . . . 48
gson_GO . . . 49
gson_GO_local . . . 49
gson_KEGG . . . 50
gson_KEGG_mapper . . . 50
gson_KO . . . 51
gson_WP . . . 52
idType . . . 52
interpret . . . 53
interpret-evidence . . . 55
interpret_agent . . . 56
interpret_dialogue . . . 59
interpret_hierarchical . . . 61
ko2name . . . 62
mechanism_alias_regulator . . . 62
mechanism_discover . . . 63
mechanism_graph . . . 65
mechanism_preflight . . . 66
mechanism_report . . . 67
mechanism_reproduction . . . 69
mechanism_sets . . . 70
mechanism_significant . . . 72
mechanism_tfunion . . . 73
mechanism_writable . . . 74
mechanism_xref . . . 75
merge_result . . . 76
mnseGO . . . 76
mnseKEGG . . . 78
mnseMKEGG . . . 79
mnseWP . . . 81
nseGO . . . 82
nseKEGG . . . 83
nseMKEGG . . . 84
nseWP . . . 85

pathway_evidence 86
plot.interpretation 88
plot_mechanism 89
plot_mechanism_tree 89
plotGOgraph 91
print.adjudication 92
print.evidenceBundle 92
provenance 93
read.gmt.pc 93
reduce_redundancy 94
say 95
search_kegg_organism 96
simplify 96
uniprot_get 98

Index **100**

`add_go_level` *Add the GO level of each enriched term*

Description

'enrichGO()'/'gseGO()' results carry the term, its ontology and its statistics, but not how deep the term sits in the GO hierarchy. This appends that as a 'level' column (1 = ontology root), which makes it easy to keep a band of levels by filtering, e.g. 'subset(res, level >= 3 & level <= 6)'.

Usage

```
add_go_level(x)
```

Arguments

x	an 'enrichResult', 'gseaResult' or 'compareClusterResult' from a GO enrichment analysis

Details

The level is taken from the ontology the term itself belongs to, so an 'ont = "ALL"' (or 'compareCluster()') result is labelled correctly term by term. Terms that GO.db does not recognise get 'NA'.

Value

'x' with a 'level' column appended to its result table

Author(s)

Guangchuang Yu https://yulab-smu.top

Examples

```
## Not run:
x <- enrichGO(gene, OrgDb = "org.Hs.eg.db", ont = "BP")
x <- add_go_level(x)
subset(x, level >= 3 & level <= 6)

## End(Not run)
```

`adjudicate` *Decide which mechanisms the evidence actually supports*

Description

Combine an ‘evidenceBundle‘ into one verdict per regulator, and return a verdict object rather than a bare table. The point is not to rank by p-value but to say which claims survive a change of evidence type, and – just as importantly – which ones were never examined.

Usage

```
adjudicate(x, min_independent = 2, coverage_floor = NULL)
```

Arguments

`x`	An ‘evidenceBundle‘ from [evidence()].
`min_independent`	Distinct sources required for ‘supported‘.
`coverage_floor`	A source whose ‘coverage_tested‘ is below this is reported as ‘untested‘ whatever it found. ‘NULL‘ takes the floor recorded in ‘x$input$design‘.

Details

Adjudication is a step. ‘evidence()‘ collects and grades, ‘adjudicate()‘ decides, [interpret()] writes. Only the last one calls a model, and it needs an ‘adjudication‘ – not a bundle – so the decision cannot be skipped on the way to prose.

The name is deliberate: this function does not find where sources agree, it decides which claims survive. "Consensus" in bioinformatics means agreement between clusterings, which is a different operation.

Four verdicts:

* ‘supported‘ – at least ‘min_independent‘ distinct sources agree **on the same entity**, and at least one of them is graded B or above. A regulator found only at level C is never ‘supported‘, however small its p-value.

Agreement is only counted for an identical regulator label. A transcription factor and a microRNA that happen to share targets are two different entities: ‘FOXO3‘ supported by TF target sets and ‘MIR23C‘ supported by miRNA target sets are each ‘single_source‘, not one ‘supported‘ finding. Cross-type agreement is a statement about the gene list, not about either regulator, and it belongs in [evidence_ppi()] or a manual comparison.

The same applies across levels of the same source family: a cell-state name such as ‘HAY_BONE_MARROW_STROMAL‘ and a tissue name such as ‘CAR_HPX‘ are entities to be compared with **each other**, never summed with a factor. Their identifiers also mean something different – "this signature resembles that state" rather than "this factor may drive these genes" – so a ‘supported‘ verdict for them would be a category error, not evidence. Both are graded C by [mechanism_xref()] for that reason, and a C-level entity is reported as single_source at best.

This has a consequence worth stating plainly. With ‘upstream = c("tf", "mirna")‘ two sources never name the same entity, so ‘supported‘ cannot occur; every finding is ‘single_source‘, ‘absent‘ or ‘untested‘. The verdict becomes reachable once evidence that names the *same* entity is added – a second target collection for the same factor, a perturbation signature naming the feature it perturbed, or a network grade attached to the same regulator. Until then ‘adjudicate()‘ still does its job: it separates what was tested and not found from what was never tested, which is the part a p-value threshold cannot do. * ‘single_source‘ – one source names it. That is a hypothesis, not a mechanism. * ‘absent‘ – the source could have spoken about these genes (its coverage reaches ‘coverage_floor‘, or no floor was set) and did not. * ‘untested‘ – the source never examined these genes. **‘absent‘ is evidence; ‘untested‘ is the absence of evidence**, and reporting the second as the first is the commonest way a regulator analysis overstates itself.

Value

An S3 object of class ‘adjudication‘ with ‘verdicts‘, ‘evidence‘, ‘diagnostics‘, ‘parameters‘ and ‘provenance‘. ‘$verdicts‘ is a ‘data.frame‘ whose columns, in order, are ‘regulator‘, ‘sources‘, ‘n_sources‘, ‘max_level‘, ‘verdict‘, ‘padj‘, ‘family‘, ‘evidence_kind‘, ‘direction‘, ‘measured_direction‘, ‘perturbation_effect‘ and ‘writable‘ (the sentence the evidence permits; see [mechanism_writable()]).

‘$provenance$evidence_sha‘ fingerprints the genes, background, normalized evidence rows, coverage, design, decision parameters and collection release – and deliberately not the time, so two runs over the same evidence hash alike and a comparison means "same evidence" rather than "same moment".

‘direction‘ and ‘measured_direction‘ answer different questions and are kept apart deliberately. ‘direction‘ is the collection’s: which way the regulator’s programme moved in the caller’s data, from the sign of the input half it was enriched in or the sign of its NES. ‘measured_direction‘ is the perturbation’s: which way genes responded when the factor was knocked down, which is ‘mixed‘ for most real knockdowns because removing a factor moves some genes up and others down. Merging the two into one column made the knockdown’s ‘mixed‘ overwrite the collection’s established sign on exactly the best-supported entities.

See Also

[evidence()], [interpret()], [mechanism_writable()]

Examples

```
## Not run:
ev <- evidence(setNames(c(2.1, 1.8, 1.4), c("PGK1", "CA9", "VEGFA")))
adj <- adjudicate(ev)
adj
consensus_findings(adj)

## End(Not run)
```

append_kegg_category *append_kegg_category*

Description

add KEGG pathway category information

Usage

```
append_kegg_category(x)
```

Arguments

x	KEGG enrichment result

Details

This function appends the KEGG pathway category information to KEGG enrichment result (either output of 'enrichKEGG' or 'gseKEGG')

The classification is the KEGG PATHWAY hierarchy published on https://www.genome.jp/kegg/pathway.html, shipped as a cached table rather than fetched on every call. A pathway released after the cache was built has no row there and comes back with NA in category and subcategory; the cache is refreshed with the package.

Value

update KEGG enrichment result with category information

Author(s)

Guangchuang Yu

ask *Send a question to an evidence dialogue*

Description

Continues the conversation opened by [interpret_dialogue()]. The reply is checked against the admitted entities before it is returned, so a model that reaches for a regulator the evidence does not contain is reported rather than believed.

Usage

```
ask(dialogue, question, ...)
```

Arguments

dialogue	An ‘evidenceDialogue‘ from [interpret_dialogue()].
question	The question to ask.
...	Ignored, for method compatibility.

Value

The updated ‘evidenceDialogue‘, with the exchange appended to ‘transcript‘ and any out-of-bounds naming recorded in the last turn’s ‘unsupported‘.

See Also

[interpret_dialogue()], [say()]

---

attribute_genes *Attribute each gene to the regulator that best explains it*

Description

Collapses the many-to-many regulator/gene relation into a tree: each input gene is assigned to exactly one regulator, and the assignment follows the evidence rather than the layout.

Collapses the many-to-many regulator/gene relation into a tree: each input gene is assigned to exactly one regulator, and the assignment follows the evidence rather than the layout.

Usage

```
attribute_genes(x)

attribute_genes(x)
```

Arguments

x	A ‘mechgraph‘ from [mechanism_graph()].

Details

A network answers "who claims this gene". A mechanism needs "who explains it", and when three regulators each list the same gene the figure cannot say which one the evidence favours – every line looks the same. This assigns one parent per gene, by a rule stated in order: strongest grade first, then the most significant set, then the largest overlap. The result is a forest that can be drawn as a tree, with the regulators at the root end and the genes at the tips.

A gene claimed by no regulator is absent, not unassigned: it is left out rather than attached to a root it has nothing to do with.

The order matters and is deliberate. Grade before significance because a measured mechanism outranks a well-powered prediction; significance before overlap because a set that barely reaches threshold should not win on size.

A network answers "who claims this gene". A mechanism needs "who explains it", and when three regulators each list the same gene the figure cannot say which one the evidence favours – every line looks the same. This assigns one parent per gene, by a rule stated in order: strongest grade first, then the most significant set, then the largest overlap. The result is a forest that can be drawn as a tree, with the regulators at the root end and the genes at the tips.

A gene claimed by no regulator is absent, not unassigned: it is left out rather than attached to a root it has nothing to do with.

The order matters and is deliberate. Grade before significance because a measured mechanism outranks a well-powered prediction; significance before overlap because a set that barely reaches threshold should not win on size.

Value

A data frame with ‘gene‘, ‘regulator‘, ‘level‘, ‘padj‘ and ‘n_claimed‘ (how many regulators listed it, which is the ambiguity this rule resolves).

A data frame with ‘gene‘, ‘regulator‘, ‘level‘, ‘padj‘ and ‘n_claimed‘ (how many regulators listed it, which is the ambiguity this rule resolves).

See Also

[mechanism_graph()], [plot_mechanism_tree()]

Examples

```
## Not run:
attribute_genes(mechanism_graph(ev))

## End(Not run)
## Not run:
attribute_genes(mechanism_graph(ev))

## End(Not run)
```

`bitr` *bitr*

Description

Biological Id TRanslator

Usage

```
bitr(geneID, fromType, toType, OrgDb, drop = TRUE)
```

Arguments

geneID	input gene id
fromType	input id type
toType	output id type
OrgDb	annotation db
drop	drop NA or not

Value

data.frame

Author(s)

Guangchuang Yu

---

bitr_kegg *bitr_kegg*

Description

convert biological ID using KEGG API

Usage

```
bitr_kegg(geneID, fromType, toType, organism, drop = TRUE)
```

Arguments

geneID	input gene id
fromType	input id type, one of "Path", "Module", "ncbi-proteinid", "ncbi-geneid", "uniprot", "kegg", and "ko"
toType	output id type, one of "Path", "Module", "ncbi-proteinid", "ncbi-geneid", "uniprot", "kegg", and "ko"
organism	supported organism, can be search using search_kegg_organism function
drop	drop NA or not

Value

data.frame

Author(s)

Guangchuang Yu

Examples

```
## Not run:
  bitr_kegg(c("P31946", "P01308"), "uniprot", "ko", "hsa")
  bitr_kegg(c("K04456", "K04526"), "ko", "uniprot", "hsa")

## End(Not run)
```

`browseKEGG` *browseKEGG*

Description

open KEGG pathway with web browser

Usage

```
browseKEGG(x, pathID)
```

Arguments

`x`	an instance of enrichResult or gseaResult
`pathID`	pathway ID

Value

url

Author(s)

Guangchuang Yu

`chat` *Chat about an evidence bundle, turn by turn*

Description

Opens an interactive conversation: the findings are printed, then each line typed is sent to the model until the caller leaves. This is the same dialogue [ask()] drives, with the loop written for you.

Usage

```
chat(dialogue, prompt = "? ", echo = TRUE)
```

Arguments

dialogue	An ‘evidenceDialogue‘ from [interpret_dialogue()].
prompt	Text shown before each input line.
echo	Print the model’s reply as it arrives. ‘FALSE‘ returns the dialogue silently, for testing.

Details

Two prefixes are commands rather than questions:

* ‘¡ records what you type as a caller assertion – something you know that the data does not. It is stored at level ‘U‘ and never changes a verdict. * ‘:q‘ leaves, returning the dialogue so the transcript can be kept.

Anything else is a question. The bounds are the same as [ask()]: a regulator the evidence does not contain is reported, not discussed.

Outside an interactive session this reads from stdin, so it also works when piping a script of questions into ‘Rscript‘.

Value

The updated ‘evidenceDialogue‘.

See Also

[interpret_dialogue()], [ask()], [say()]

Examples

```
## Not run:
ev <- evidence(genes, upstream = c("tf", "perturbagen"),
               design = list(treatment = "knockdown"))
chat(interpret_dialogue(ev, context = "airway smooth muscle"))

## End(Not run)
```

---

collapse_findings *Collapse redundant findings before reporting them*

Description

Groups findings whose gene sets largely overlap and keeps the strongest member of each group, so a result reads as a handful of statements rather than as every set that crossed a threshold.

Usage

```
collapse_findings(x, threshold = 0.5, species = NULL, release = NULL)

## S3 method for class 'collapsedFindings'
print(x, ...)
```

Arguments

x	An object to print.
threshold	Jaccard index above which two findings are the same statement. 0.5 means "shares most of its genes".
species	Species whose collections supply the gene sets.
release	MSigDB release, taken from the bundle when ‘NULL‘.
...	Ignored.

Details

The need is not uniform across sources, which is why this is a separate step rather than something [adjudicate()] always does. Measured on the airway list: transcription-factor findings barely overlap at all (median pairwise Jaccard 0.000 over 155 findings), cell-state findings likewise (0.008 over 337), and only tissue signatures are genuinely redundant (17 collapsing them keeps 17 of 42). Running a collapse step that is not needed discards findings that say different things.

Value

A data frame like ‘consensus_findings()‘, with two extra columns: ‘collapsed‘ (how many findings this row stands for) and ‘absorbed‘ (the labels it replaced). Ordered by significance.

‘x‘, invisibly.

See Also

[adjudicate()], [consensus_findings()]

Examples

```
## Not run:
ev <- evidence(genes, upstream = "tissue")
collapse_findings(ev)

## End(Not run)
```

---

compareCluster *Compare gene clusters functional profile*

Description

Given a list of gene set, this function will compute profiles of each gene cluster.

Usage

```
compareCluster(
  geneClusters,
  fun = "enrichGO",
  data = "",
  source_from = NULL,
  ...
)
```

Arguments

geneClusters	a list of entrez gene id. Alternatively, a formula of type Entrez~group or a formula of type Entrez | logFC ~ group for "gseGO", "gseKEGG" and "GSEA".
fun	One of "groupGO", "enrichGO", "enrichKEGG", "enrichDO" or "enrichPathway" . Users can also supply their own function.
data	if geneClusters is a formula, the data from which the clusters must be extracted.
source_from	If using a custom function in "fun", provide the source package as a string here. Otherwise, the function will be obtained from the global environment.
...	Other arguments.

Value

A clusterProfResult instance.

Author(s)

Guangchuang Yu https://yulab-smu.top

See Also

[compareClusterResult-class], [groupGO], [enrichGO], [enrichKEGG], [enrichDO][DOSE::enrichDO], [enrichPathway][ReactomePA::enrichPathway]

Examples

```
## Not run:
data(gcSample)
xx <- compareCluster(gcSample, fun="enrichKEGG",
                     organism="hsa", pvalueCutoff=0.05)
as.data.frame(xx)
# plot(xx, type="dot", caption="KEGG Enrichment Comparison")
dotplot(xx)

## formula interface
mydf <- data.frame(Entrez=c('1', '100', '1000', '100101467',
                            '100127206', '100128071'),
                   logFC = c(1.1, -0.5, 5, 2.5, -3, 3),
                   group = c('A', 'A', 'A', 'B', 'B', 'B'),
                   othergroup = c('good', 'good', 'bad', 'bad', 'good', 'bad'))
```

```
xx.formula <- compareCluster(Entrez~group, data=mydf,
                             fun='groupGO', OrgDb='org.Hs.eg.db')
as.data.frame(xx.formula)

## formula interface with more than one grouping variable
xx.formula.twogroups <- compareCluster(Entrez~group+othergroup, data=mydf,
                                       fun='groupGO', OrgDb='org.Hs.eg.db')
as.data.frame(xx.formula.twogroups)


## End(Not run)
```

`consensus_findings` *The findings an adjudication allows you to talk about*

Description

Keep the entities whose verdict is ‘supported‘ or ‘single_source‘, dropping the ones that were examined and found wanting. This is the set a narrative may discuss, and it is exported because the alternative – filtering on the verdict column by hand – is easy to get subtly wrong: ‘absent‘ is evidence against, ‘untested‘ is no evidence at all, and neither belongs in a claim.

Usage

```
consensus_findings(cv)
```

Arguments

`cv`	An ‘adjudication‘ from [adjudicate()], or its ‘$verdicts‘ table.

Value

A ‘data.frame‘ of the admitted rows.

See Also

[adjudicate()], [collapse_findings()], [interpret()]

Examples

```
## Not run:
consensus_findings(adjudicate(ev))

## End(Not run)
```

DataSet *Datasets gcSample contains a sample of gene clusters.*

Description

Datasets gcSample contains a sample of gene clusters.

Datasets kegg_species contains kegg species information

Datasets kegg_category contains kegg pathway category information

Datasets DE_GSE8057 contains differential epxressed genes obtained from GSE8057 dataset

download_KEGG *download_KEGG*

Description

download the latest version of KEGG pathway/module

Usage

```
download_KEGG(species, keggType = "KEGG", keyType = "kegg")
```

Arguments

species	species
keggType	one of 'KEGG' or 'MKEGG'
keyType	supported keyType, see bitr_kegg

Value

list

Author(s)

Guangchuang Yu

dropGO *dropGO*

Description

drop GO term of specific level or specific terms (mostly too general).

Usage

```
dropGO(x, level = NULL, term = NULL)
```

Arguments

x	an instance of 'enrichResult' or 'compareClusterResult'
level	GO level
term	GO term

Value

modified version of x

Author(s)

Guangchuang Yu

---

enrichDAVID *enrichDAVID*

Description

enrichment analysis by DAVID

Usage

```
enrichDAVID(
  gene,
  idType = "ENTREZ_GENE_ID",
  universe,
  minGSSize = 10,
  maxGSSize = 500,
  annotation = "GOTERM_BP_FAT",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  qvalueCutoff = 0.2,
  species = NA,
  david.user
)
```

Arguments

gene	input gene
idType	id type
universe	background genes. If missing, the all genes listed in the database (eg TERM2GENE table) will be used as background.
minGSSize	minimal size of genes annotated for testing
maxGSSize	maximal size of genes annotated for testing
annotation	david annotation
pvalueCutoff	adjusted pvalue cutoff on enrichment tests to report
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
qvalueCutoff	qvalue cutoff on enrichment tests to report as significant. Tests must pass i) pvalueCutoff on unadjusted pvalues, ii) pvalueCutoff on adjusted pvalues and iii) qvalueCutoff on qvalues to be reported.
species	species
david.user	david user

Value

A enrichResult instance

Author(s)

Guangchuang Yu

---

enricher *enricher*

Description

A universal enrichment analyzer

Usage

```
enricher(
  gene,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  universe = NULL,
  minGSSize = 10,
  maxGSSize = 500,
  qvalueCutoff = 0.2,
  gson = NULL,
  TERM2GENE,
  TERM2NAME = NA
)
```

Arguments

gene	a vector of gene id
pvalueCutoff	adjusted pvalue cutoff on enrichment tests to report
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
universe	background genes. If missing, the all genes listed in the database (eg TERM2GENE table) will be used as background.
minGSSize	minimal size of genes annotated for testing
maxGSSize	maximal size of genes annotated for testing
qvalueCutoff	qvalue cutoff on enrichment tests to report as significant. Tests must pass i) pvalueCutoff on unadjusted pvalues, ii) pvalueCutoff on adjusted pvalues and iii) qvalueCutoff on qvalues to be reported.
gson	a GSON object, if not NULL, use it as annotation data.
TERM2GENE	user input annotation of TERM TO GENE mapping, a data.frame of 2 column with term and gene. Only used when gson is NULL.
TERM2NAME	user input of TERM TO NAME mapping, a data.frame of 2 column with term and name. Only used when gson is NULL.

Value

A enrichResult instance

Author(s)

Guangchuang Yu https://yulab-smu.top

---

enrichGO *GO Enrichment Analysis of a gene set. Given a vector of genes, this function will return the enrichment GO categories after FDR control.*

Description

GO Enrichment Analysis of a gene set. Given a vector of genes, this function will return the enrichment GO categories after FDR control.

Usage

```
enrichGO(
  gene,
  OrgDb,
  keyType = "ENTREZID",
  ont = "MF",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  universe,
```

```
  qvalueCutoff = 0.2,
  minGSSize = 10,
  maxGSSize = 500,
  readable = FALSE,
  pool = FALSE,
  evidence = NULL
)
```

Arguments

gene	a vector of entrez gene id.
OrgDb	OrgDb
keyType	keytype of input gene
ont	One of "BP", "MF", and "CC" subontologies, or "ALL" for all three.
pvalueCutoff	adjusted pvalue cutoff on enrichment tests to report
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
universe	background genes. If missing, the all genes listed in the database (eg TERM2GENE table) will be used as background.
qvalueCutoff	qvalue cutoff on enrichment tests to report as significant. Tests must pass i) pvalueCutoff on unadjusted pvalues, ii) pvalueCutoff on adjusted pvalues and iii) qvalueCutoff on qvalues to be reported.
minGSSize	minimal size of genes annotated by Ontology term for testing.
maxGSSize	maximal size of genes annotated for testing
readable	whether mapping gene ID to gene Name
pool	If ont='ALL', whether pool 3 GO sub-ontologies
evidence	optional character vector of GO evidence codes to keep (for example 'c("IDA", "IPI", "IMP")', or '"IEA"' for electronic annotations only). When supplied, annotations carrying any other evidence code are dropped before the analysis — useful to avoid circular reasoning (dropping 'IEP' when the gene sets come from co-expression clustering) or to keep only manually reviewed annotations (dropping 'IEA'). Because the annotation changes, so does the background, so the resulting terms are not simply a subset of the unfiltered run. Filtered runs are not cached, so they rebuild the annotation each time. 'NULL' (default) uses every annotation.

Value

An enrichResult instance.

Author(s)

Guangchuang Yu https://yulab-smu.top

See Also

[enrichResult-class], [compareCluster]

Examples

```
## Not run:
  data(geneList, package = "DOSE")
 de <- names(geneList)[1:100]
 yy <- enrichGO(de, 'org.Hs.eg.db', ont="BP", pvalueCutoff=0.01)
 head(yy)

## End(Not run)
```

---

enrichKEGG *KEGG Enrichment Analysis of a gene set. Given a vector of genes, this function will return the enrichment KEGG categories with FDR control.*

Description

KEGG Enrichment Analysis of a gene set. Given a vector of genes, this function will return the enrichment KEGG categories with FDR control.

Usage

```
enrichKEGG(
  gene,
  organism = "hsa",
  keyType = "kegg",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  universe,
  minGSSize = 10,
  maxGSSize = 500,
  qvalueCutoff = 0.2,
  use_internal_data = FALSE
)
```

Arguments

gene	a vector of entrez gene id.
organism	supported organism listed in 'https://www.genome.jp/kegg/catalog/org_list.html'; alternatively a 'GSON' object (e.g. from 'gson_KEGG()') holding the annotation, which allows the analysis to run without contacting KEGG
keyType	one of "kegg", 'ncbi-geneid', 'ncbi-proteinid' and 'uniprot'
pvalueCutoff	adjusted pvalue cutoff on enrichment tests to report
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
universe	background genes. If missing, the all genes listed in the database (eg TERM2GENE table) will be used as background.
minGSSize	minimal size of genes annotated by Ontology term for testing.

maxGSSize	maximal size of genes annotated for testing
qvalueCutoff	qvalue cutoff on enrichment tests to report as significant. Tests must pass i) pvalueCutoff on unadjusted pvalues, ii) pvalueCutoff on adjusted pvalues and iii) qvalueCutoff on qvalues to be reported.
use_internal_data	logical, use KEGG.db or latest online KEGG data

Value

A enrichResult instance.

Author(s)

Guangchuang Yu https://yulab-smu.top

See Also

[enrichResult-class], [compareCluster]

Examples

```
## Not run:
  data(geneList, package='DOSE')
  de <- names(geneList)[1:100]
  yy <- enrichKEGG(de, pvalueCutoff=0.01)
  head(yy)

## End(Not run)
```

---

enrichMKEGG *KEGG Module Enrichment Analysis of a gene set. Given a vector of genes, this function will return the enrichment KEGG Module categories with FDR control.*

Description

As with enrichKEGG(), organism may be a species code or a GSON object, so module enrichment can be run against a locally built annotation (see gson_KEGG()) when KEGG is unreachable.

Usage

```
enrichMKEGG(
  gene,
  organism = "hsa",
  keyType = "kegg",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  universe,
```

```
  minGSSize = 10,
  maxGSSize = 500,
  qvalueCutoff = 0.2
)
```

Arguments

gene	a vector of entrez gene id.
organism	supported organism listed in 'https://www.genome.jp/kegg/catalog/org_list.html'; alternatively a 'GSON' object (e.g. from 'gson_KEGG()') holding the annotation, which allows the analysis to run without contacting KEGG
keyType	one of "kegg", 'ncbi-geneid', 'ncbi-proteinid' and 'uniprot'
pvalueCutoff	adjusted pvalue cutoff on enrichment tests to report
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
universe	background genes. If missing, the all genes listed in the database (eg TERM2GENE table) will be used as background.
minGSSize	minimal size of genes annotated by Ontology term for testing.
maxGSSize	maximal size of genes annotated for testing
qvalueCutoff	qvalue cutoff on enrichment tests to report as significant. Tests must pass i) pvalueCutoff on unadjusted pvalues, ii) pvalueCutoff on adjusted pvalues and iii) qvalueCutoff on qvalues to be reported.

Value

A enrichResult instance.

---

enrichPC *enrichPC*

Description

ORA analysis for Pathway Commons

Usage

```
enrichPC(gene, ...)
```

Arguments

gene	a vector of genes (either hgnc symbols or uniprot IDs)
...	additional parameters, see also the parameters supported by the enricher() function

Details

This function performs over-representation analysis using Pathway Commons

Value

A enrichResult instance

---

enrichWP *enrichWP*

Description

ORA analysis for WikiPathways

Usage

```
enrichWP(gene, organism, ...)
```

Arguments

gene	a vector of entrez gene id
organism	supported organisms, which can be accessed via the get_wp_organisms() function
...	additional parameters, see also the parameters supported by the enricher() function

Details

This function performs over-representation analysis using WikiPathways

Value

A enrichResult instance

Author(s)

Guangchuang Yu

`evidence` *Grade upstream evidence for a gene list*

Description

Turn a gene list into a table of candidate mechanisms, each carrying an evidence level that says how strongly it may be stated. This is the mechanism-exit counterpart of a pathway enrichment: the collections tested name *upstream participants* (transcription factors, microRNAs, ...) instead of biological processes, so an enriched set answers "what could have produced these genes" rather than "what do these genes do".

Usage

```
evidence(
  genes,
  design = list(),
  pathway = FALSE,
  upstream = "tf",
  universe = NULL,
  network = FALSE,
  minGSSize = 5,
  maxGSSize = Inf,
  tft = NULL,
  print_pvalue = 1,
  print_qvalue = 1
)
```

Arguments

`genes` Gene symbols, or a **named numeric vector** of gene symbols and their statistics. A named numeric is tested by direction: the positive half and the negative half are run separately against each collection, so a regulator's row carries whether its programme rose or fell. Measured on the hypoxia contrast this took ‘HIF1_Q3‘ from padj 1.2e-4 (both directions mixed into one list) to 1.0e-5, because asking "which set is enriched among what changed" dilutes the signal when half the changes go the other way. Both halves are corrected together, since splitting doubles the number of hypotheses. An ‘enrichResult‘ (its ‘geneID‘ column is expanded), a ‘gseaResult‘ (the union of its ‘core_enrichment‘ genes, with the ranked list as the background), or a ‘data.frame‘ with a column named by ‘design$gene_col‘.

A ‘gseaResult‘ contributes the genes that produced each enriched set's ranking rather than every gene in ‘@geneList‘: asking which regulator explains all of a transcriptome has no answer, and on the airway list that form returned no ‘supported‘ regulator at all. Measured against known answers, the GSEA form and the ‘enrichResult‘ form rank the true regulator identically (MRR 0.046 against 0.047), because the limit is the over-representation test's power at small overlaps, not the input.

`design` A list describing the experiment; see Details.

`pathway` The conventional enrichment layer. ‘FALSE‘ (the default) skips it. ‘TRUE‘ runs it here, on the up and the down half separately, and then asks which regulator’s target set explains each pathway – a different question from the collection layer, which asks which regulator’s target set is over-represented in the whole list. An ‘enrichResult‘ or ‘gseaResult‘ reuses a result the caller already built instead of recomputing it, which is also the way to pass one: ‘evidence()‘ no longer takes a result as ‘genes‘, because reading genes back out of it silently drops every gene that appears only in a pathway below its p-value cutoff.

The pathway rows are kept in ‘$pathways‘, never in ‘$regulators‘. They are derived from the same collections, so counting them as a source would let one collection corroborate itself.

`upstream` Collections to test: ‘"tf"‘, ‘"tfunion"‘, ‘"mirna"‘, ‘"perturbagen"‘, ‘"cellstate"‘ or ‘"tissue"‘. Defaults to ‘"tf"‘ alone.

‘"tfunion"‘ is C3:TFT plus four public transcription-factor libraries (ENCODE, ChEA, TF_Perturbations and TRRUST), which together name 315 factors C3:TFT has no set for – including ‘EPAS1‘, ‘NFKB1‘, ‘TP53‘, ‘CTCF‘, ‘ESR1‘, ‘JUN‘ and ‘FOS‘. Prefer it when the question is "which regulator drives this list" rather than "which of these specific motif families is enriched": on a hypoxia contrast it recovered the paper’s answer (HIF-2) where ‘"tf"‘ could not, at 2869 sets against 1114, and three random gene sets of the same size still returned no significant set. See [mechanism_tfunion()].

Pass it **together with a measured source**, e.g. ‘c("tfunion", "perturbagen")‘. The union is one source however many sets it holds, and corroboration counts sources rather than rows, so on its own it can only reach ‘single_source‘ – a page of those reads like a failure when the finding is simply uncorroborated.

‘"mirna"‘ is off by default for a measured reason: its sets are predicted seed matches, so an enrichment describes which genes carry the motif rather than which microRNA is active. Measured on the airway list, 7 miRNAs were significant among the UP-regulated genes and 0 among the down ones – the opposite of what a repressor predicts – so the direction that would make the finding interpretable is absent. Ask for it explicitly when the question is about UTR composition.

`universe` Background genes. ‘NULL‘ uses every symbol in ‘org.Hs.eg.db‘.

`network` Whether to query STRING for interaction evidence. Off by default because it needs the network.

The result is reference material, not a vote. ‘$ppi‘ lists the edges and grades them (see [evidence_ppi()]), but nothing turns an edge into a corroboration, so ‘network = TRUE‘ changes no verdict. That is a deliberate limit rather than an oversight: measured on 54 datasets whose true regulator is known, the network contained it for 35 and moved no answer. Set it when a reader will look at the interactions; do not expect it to support a ‘supported‘.

`minGSSize` Minimum gene-set size. The default is 5, not ‘enricher()‘’s 10, so that small motif sets are tested; ‘maxGSSize‘ is ‘Inf‘ because the largest GTRD target sets carry more than a thousand genes and the default upper bound of 500 would drop exactly the broad regulators that matter.

`maxGSSize` Maximum gene-set size.

`tft` Which part of the transcription-factor collection to test. ‘NULL‘ (the default) uses all of C3:TFT. ‘"gtrd"‘ keeps only the ChIP-defined sets, whose identifiers are gene symbols: measured on the airway list that takes the share of labels naming a family or a consensus sequence from 40 and those labels are the ones no perturbation dataset can ever answer for. ‘"motif"‘ selects the predicted half instead; a character vector selects several, e.g. ‘c("gtrd", "motif")‘.

Only ‘upstream = "tf"‘ honours this. Asking for it together with ‘"tfunion"‘ is an error rather than a silent no-op: the union’s other four libraries have no GTRD/motif split, so ignoring the filter would answer a wider question than the one asked.

`print_pvalue`, `print_qvalue`

Cutoffs used by the print method only. They do not filter the result: ‘$regulators‘ always holds every tested set, so that a set filtered out for display can still be distinguished from a set that was never tested.

Details

Each requested collection is tested separately against the same gene list, and each is corrected within itself. Pooling them would let a large collection dilute a small one: C3:TFT holds about 1100 sets and C3:MIR about 2600, with different target definitions, so their p-values are not on one scale. ‘$regulators‘ therefore carries ‘padj_within_source‘, never a single shared ‘padj‘.

Every returned collection must be graded the same way: an over-represented target set is evidence that the regulator’s *fingerprint* sits in the list, not evidence that the regulator caused it. ‘$regulators$level‘ says how far a claim may go, and ‘$diagnostics‘ says how much of the list the collection could speak about at all.

‘design‘ accepts ‘contrast‘, ‘system‘, ‘species‘, ‘treatment‘ (‘"none"‘ by default), ‘gene‘, ‘direction‘, ‘gene_col‘ and ‘coverage_floor‘. ‘treatment‘ decides whether perturbation evidence (level A) is reachable; it is never inferred from free text. ‘background‘ names entities that belong to the setting rather than to the findings – the receptor a drug acts through, a known positive control – so that ‘unsupported_claims‘ does not report them as unevidenced. ‘coverage_floor‘ is ‘NULL‘ by default: a collection is only declared ‘untested‘ as a whole when the caller states a floor.

Value

An ‘evidenceBundle‘: a list with ‘input‘, ‘sets‘, one element per tested collection (‘tf‘, ‘mirna‘), ‘regulators‘, ‘xref‘, ‘diagnostics‘ and ‘provenance‘.

‘diagnostics‘ reports counts whose denominators differ and must not be conflated: ‘n_input‘ is the number of genes handed in; ‘n_tested‘ is how many of those could enter the test at all – inputs present in the universe and in some set, which is the ‘DESize‘ that ‘enrichR‘’s ‘GeneRatio‘ divides by; ‘n_sig_covered‘ is how many are covered by a *significant* regulator; ‘uncovered_sig‘ is ‘n_input - n_sig_covered‘. ‘coverage_tested‘ is ‘n_tested / n_input‘, not the larger ratio an "appears in any set" count would give.

See Also

[mechanism_sets()] for the collections, [mechanism_xref()] for the level rules.

Examples

```
## Not run:
ev <- evidence(c("HIF1A", "PGK1", "CA9", "BNIP3", "SLC2A1", "LDHA"))
ev$regulators

## End(Not run)
```

`evidence_biotrigger` *Upstream regulators from GPSAdb's biotrigger*

Description

Ask which perturbed genes move the caller's list, which is the reverse question to [evidence_similarity()]: instead of "which experiment looks like this list" it asks "perturbing what changes this list".

Usage

```
evidence_biotrigger(
  genes,
  effect = c("both", "activated", "inhibited"),
  limit = 20,
  min_genes = 5
)
```

Arguments

`genes`	Gene symbols.
`effect`	Restrict to ‘"activated"‘, ‘"inhibited"‘ or ‘"both"‘.
`limit`	Number of hits.
`min_genes`	Minimum list size worth asking about.

Details

A biotrigger hit is a measured perturbation whose effect overlaps the list, so it is graded **B**, the same as a GTRD target set: the factor was measured, but the measurement is of a signature, not of binding, and the overlap of two signatures is weaker than a direct assay of the same factor. It is not graded C, because unlike a motif it is not a prediction.

The genes returned are *perturbed genes*, not necessarily regulators. A hit on a metabolic enzyme means perturbing that enzyme shifts the list, which is a downstream relationship. The returned rows carry the effect direction so the caller can judge, and a hit is not a claim that the gene regulates anything.

Value

A ‘data.frame‘ with ‘regulator‘, ‘source‘, ‘term‘, ‘named‘, ‘level‘, ‘padj_within_source‘, ‘setSize‘, ‘Count‘, ‘family‘, ‘evidence_kind‘, ‘direction‘ and the dataset columns, in the shape [evidence()] uses.

See Also

[evidence_similarity()], [evidence()]

Examples

```
## Not run:
evidence_biotrigger(c("HIF1A", "PGK1", "CA9", "BNIP3", "LDHA"))

## End(Not run)
```

`evidence_perturbation` *Perturbation evidence from GPSAdb*

Description

Ask GPSAdb which measured perturbation moves the input genes, and grade the result. A perturbation dataset whose target set is enriched among the input genes is the only evidence that can reach level **A**: it is an experiment, not an annotation or a prediction.

Usage

```
evidence_perturbation(
  genes,
  method = NULL,
  limit = 50,
  min_genes = 5,
  pvalueCutoff = 0.05,
  candidates = NULL,
  considered = NULL,
  system = NULL,
  up = NULL,
  down = NULL
)
```

Arguments

`genes`	Gene symbols.
`method`	Perturbation method to query, e.g. ‘"knockdown"‘, ‘"knockout"‘, ‘"overexpression"‘, ‘"drug"‘. ‘NULL‘ queries every method.
`limit`	Maximum number of datasets considered.
`min_genes`	Minimum number of input genes a dataset must contain before it is tested. A dataset with two hits says nothing about the list.
`pvalueCutoff`	Cutoff for a dataset to be reported as perturbing.

`candidates` Genes to query as the perturbed gene. Supply the regulators the collections already nominated and the sweep becomes targeted: asking whether a known candidate was ever knocked down is a question GPSAdb answers directly, while sweeping a sample of an arbitrary gene list only reports which genes happen to have datasets. Given NULL the function falls back to a spread sample and records how much it scanned in attr(x, "genes_scanned").

`considered` How many candidate regulators were eligible, when the caller already knows. Recorded on the result so a handful of datasets reads as "3 of the 30 strongest candidates had one" rather than "3 factors drive this".

`system` The biological system the datasets should come from, e.g. ‘"smooth muscle"‘ or ‘"HASM"‘. Matched against each dataset's cell line and tissue, and used only to order the evidence: a dataset from the caller's system wins the per-regulator deduplication over a more significant one from elsewhere, and every row carries ‘in_system‘ so the match (or its absence) is visible rather than assumed. Set through ‘design$system‘.

`up, down` The two halves of the caller's own contrast, as gene symbols. When given, ‘direction‘ is decided by a binomial test on the relevant half rather than by requiring every moved gene to go the same way – a knockdown necessarily moves some genes up and some down, so that requirement was never met by real data and every dataset came back ‘mixed‘. Restricted to one half, the question has an answer: among my up genes that this perturbation moves, do they move the way an activator's targets would? Both halves' counts and corrected p-values travel with the row, because which half is the right one to read depends on the direction the collection layer found, which is not known here. ‘NULL‘ (the default) keeps the older rule.

Details

The returned rows carry the same ‘regulator‘ label as the collections built by [mechanism_sets()], so a transcription factor named both by its target sets and by a knockdown dataset is *one* entity supported by *two* sources – which is what [adjudicate()] needs in order to report ‘supported‘ at all.

GPSAdb is reached through its MCP endpoint, so ‘gpsadb‘ and ‘aisdk.mcp‘ must be installed. They are suggested, not required: without them this function aborts with an install hint rather than returning an empty table, because an empty table would read as "no perturbation evidence exists".

Value

A ‘data.frame‘ with ‘regulator‘ (the perturbed gene), ‘source‘ (‘"perturbagen"‘), ‘term‘ (the dataset id), ‘named‘, ‘level‘ (‘"A"‘ when reported, ‘"B"‘ when the dataset is absent but the gene is in GPSAdb), ‘padj_within_source‘, ‘setSize‘, ‘Count‘ and ‘direction‘ (‘"up"‘, ‘"down"‘ or ‘"mixed"‘, from the sign of the responding genes).

See Also

[evidence()], [adjudicate()]

Examples

```
## Not run:
evidence_perturbation(c("HIF1A", "PGK1", "CA9"), method = "knockdown")

## End(Not run)
```

`evidence_ppi` *Network evidence for a gene list*

Description

Ask STRING how the genes relate and grade every edge by the channel that supports it. An edge is **B** when it rests on experiment (‘escore‘) or a curated database (‘dscore‘), and **C** when only text mining or co-expression supports it – a network can never produce an **A** grade, because no STRING channel measures a perturbation.

Usage

```
evidence_ppi(genes, species = "auto", required_score = NULL, add_nodes = 0)
```

Arguments

`genes`	Gene symbols.
`species`	NCBI taxonomy id, defaulting to human (9606). getPPI() can only infer a species from an enrichment result, not from a gene vector.
`required_score`	STRING’s combined-score cutoff, 0-1000.
`add_nodes`	Number of interaction partners to add.

Details

‘pscore‘ is deliberately not used for grading. It is near-zero for almost every edge, including pairs with published physical interaction (NFKB1 and NFKBIA report ‘pscore = 0‘ with ‘escore = 0.999‘), so a ‘pscore‘ threshold would grade a known physical interaction as an association.

Value

A ‘data.frame‘ with one row per edge: ‘from‘, ‘to‘, ‘score‘, ‘escore‘, ‘dscore‘, ‘tscore‘, ‘pscore‘, ‘channel‘ (the strongest channel supporting the edge) and ‘level‘ (‘"B"‘ or ‘"C"‘). An empty table means STRING returned no edge, which is a result, not an error.

See Also

[evidence()]

`evidence_status` *Evidence status of every gene named in an interpretation*

Description

Deterministically classifies each gene symbol that an interpretation names as `observed` (a member gene of the enrichment result it was given) or `out_of_evidence` (a valid symbol that is not). No model is called. Use it to label hypotheses from `interpret_agent()` as inferred rather than observed: unlike a second model, this check is exact and cannot also mark grounded genes as uncertain.

Usage

```
evidence_status(
  x,
  enrichment,
  OrgDb = NULL,
  symbols = NULL,
  fields = c("overview", "key_mechanisms", "regulatory_drivers", "crosstalk",
    "hypothesis", "narrative", "reasoning", "network_evidence")
)
```

Arguments

`x`	An `interpretation` or `interpretation_list` returned by `interpret()`, `interpret_agent()` or `interpret_hierarchical()`.
`enrichment`	The `enrichResult`, `compareClusterResult`, `gseaResult` or data.frame the interpretation was produced from (readable, so that member genes are symbols).
`OrgDb`	Optional `OrgDb` used to recognise out-of-evidence symbols. Without it, only tokens that look like gene symbols and appear in `symbols` are considered.
`symbols`	Optional character vector of valid gene symbols (alternative to `OrgDb`).
`fields`	Record fields to scan.

Value

A data.frame with columns `cluster`, `field`, `gene` and `status`.

fork_dialogue *Fork a dialogue to explore a side question*

Description

Copies a dialogue so the branch can be taken somewhere else without losing the original.

Usage

```
fork_dialogue(dialogue, name = NULL)
```

Arguments

dialogue	An ‘evidenceDialogue‘.
name	Optional label for the branch.

Details

This exists because the underlying chat session is a reference object: in ‘d <- ask(d, "q")‘ the variable is rebound but the *session* is mutated in place, so two dialogue objects that came from the same [interpret_dialogue()] share one conversation. Without a fork, asking a speculative question and then wanting to go back is not possible – the speculation is already in the history the next reply sees.

Value

A new ‘evidenceDialogue‘ with its own session, whose history starts at the point the fork was taken.

See Also

[interpret_dialogue()], [ask()]

get_ppi_network *get_ppi_network*

Description

get_ppi_network

Usage

```
get_ppi_network(
  taxID = 9606,
  version = "12.0",
  score_threshold = 400,
  output = "data.frame"
)
```

Arguments

taxID	NCBI taxon identifier (e.g., 9606 for Human).
version	STRING database version (default: "12.0").
score_threshold	threshold of significance to include a interaction (default: 400).
output	one of ‘data.frame‘ or ‘igraph‘

Details

Download and cache the full STRING PPI background network for a specific species.

Value

a ‘data.frame‘ or an ‘igraph‘ object containing the background PPI network.

---

get_wp_organisms *get_wp_organism*

Description

list supported organism of WikiPathways

Usage

```
get_wp_organisms()
```

Details

This function extracts information from ’https://data.wikipathways.org/current/gmt/’ and lists all supported organisms

Value

supported organism list

Author(s)

Guangchuang Yu

getPPI *getPPI*

Description

getPPI

Usage

```
getPPI(
  x,
  ID = 1,
  taxID = "auto",
  required_score = NULL,
  network_type = "functional",
  add_nodes = 0,
  show_query_node_labels = 0,
  output = "igraph"
)

get_ppi(
  x,
  ID = 1,
  taxID = "auto",
  required_score = NULL,
  network_type = "functional",
  add_nodes = 0,
  show_query_node_labels = 0,
  output = "igraph"
)
```

Arguments

x	an ‘enrichResult“ object or a vector of proteins, e.g. ‘c("PTCH1", "TP53", "BRCA1", "BRCA2")‘
ID	ID or index to extract genes in the enriched term(s) if ‘x‘ is an ‘enrichResult‘ object
taxID	NCBI taxon identifiers (e.g. Human is 9606, see: [STRING organisms](https://string-db.org/cgi/input.pl?input_page_active_form=organisms).
required_score	threshold of significance to include a interaction, a number between 0 and 1000 (default depends on the network)
network_type	network type: functional (default), physical
add_nodes	adds a number of proteins with to the network based on their confidence score (default:1)

show_query_node_labels
: when available use submitted names in the preferredName column when (0 or 1) (default:0)

output
: one of ‘data.frame‘ or ‘igraph‘

Details

[Getting the STRING network interactions](https://string-db.org/cgi/help.pl?sessionId=btsvnCeNrBk7).

Value

a ‘data.frame‘ or an ‘igraph‘ object

Author(s)

Yonghe Xia and modified by Guangchuang Yu

---

getTaxID *getTaxID*

Description

Convert species scientific name to taxonomic ID

Usage

```
getTaxID(species)
```

Arguments

species
: scientific name of a species

Value

taxonomic ID

Author(s)

Guangchuang Yu

getTaxInfo *getTaxInfo*

Description

Query taxonomy information from ‘stringdb‘ or ‘ensembl‘ web services

Usage

```
getTaxInfo(species, source = "stringdb")
```

Arguments

species	scientific name of a species
source	one of ‘stringdb‘ or ‘ensembl‘

Value

a ‘data.frame‘ of query information

Author(s)

Guangchuang Yu

Gff2GeneTable *Gff2GeneTable*

Description

read GFF file and build gene information table

Usage

```
Gff2GeneTable(gffFile, compress = TRUE)
```

Arguments

gffFile	GFF file
compress	compress file or not

Details

given a GFF file, this function extracts information from it and save it in working directory

Value

file save.

Author(s)

Yu Guangchuang

`go2ont` *go2ont*

Description

convert goid to ontology (BP, CC, MF)

Usage

```
go2ont(goid)
```

Arguments

`goid`	a vector of GO IDs

Value

data.frame

Author(s)

Guangchuang Yu

`go2term` *go2term*

Description

convert goid to descriptive term

Usage

```
go2term(goid)
```

Arguments

`goid`	a vector of GO IDs

Value

data.frame

Author(s)

Guangchuang Yu

`gofilter` *gofilter*

Description

filter GO enriched result at specific level

Usage

```
gofilter(x, level = 4)
```

Arguments

`x`	output from enrichGO or compareCluster
`level`	GO level

Value

updated object

Author(s)

Guangchuang Yu

`groupGO` *Functional Profile of a gene set at specific GO level. Given a vector of genes, this function will return the GO profile at a specific level.*

Description

Functional Profile of a gene set at specific GO level. Given a vector of genes, this function will return the GO profile at a specific level.

Usage

```
groupGO(
  gene,
  OrgDb,
  keyType = "ENTREZID",
  ont = "CC",
  level = 2,
  readable = FALSE
)
```

Arguments

gene	a vector of entrez gene id.
OrgDb	OrgDb
keyType	key type of input gene
ont	One of "MF", "BP", and "CC" subontologies.
level	Specific GO Level.
readable	if readable is TRUE, the gene IDs will mapping to gene symbols.

Value

A groupGOResult instance.

Author(s)

Guangchuang Yu https://yulab-smu.top

See Also

[groupGOResult-class], [compareCluster]

Examples

```
## Not run:
 data(gcSample)
 yy <- groupGO(gcSample[[1]], 'org.Hs.eg.db', ont="BP", level=2)
 head(summary(yy))
 #plot(yy)

## End(Not run)
```

`groupGOResult-class` *Class "groupGOResult" This class represents the result of functional Profiles of a set of gene at specific GO level.*

Description

Class "groupGOResult" This class represents the result of functional Profiles of a set of gene at specific GO level.

Slots

`result` GO classification result

`ontology` Ontology

`level` GO level

`organism` one of "human", "mouse" and "yeast"

`gene` Gene IDs

`readable` logical flag of gene ID in symbol or not.

Author(s)

Guangchuang Yu https://yulab-smu.top

See Also

[compareClusterResult], [compareCluster], [groupGO]

`GSEA` *GSEA*

Description

a universal gene set enrichment analysis tools

Usage

```
GSEA(
  geneList,
  exponent = 1,
  minGSSize = 10,
  maxGSSize = 500,
  eps = 1e-10,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  gson = NULL,
```

```
  TERM2GENE,
  TERM2NAME = NA,
  verbose = TRUE,
  nPerm = 1000,
  method = "multilevel",
  adaptive = FALSE,
  minPerm = 101,
  maxPerm = 1e+05,
  pvalThreshold = 0.1,
  seed = FALSE,
  ...
)
```

Arguments

geneList	order ranked geneList
exponent	weight of each step
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
eps	boundary for calculating the p value in multilevel mode
pvalueCutoff	p-value cutoff applied to both the raw p-value and the adjusted p-value (p.adjust), consistent with the historical clusterProfiler/DOSE behavior
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
gson	a GSON object, if not NULL, use it as annotation data.
TERM2GENE	user input annotation of TERM TO GENE mapping, a data.frame of 2 column with term and gene. Only used when gson is NULL.
TERM2NAME	user input of TERM TO NAME mapping, a data.frame of 2 column with term and name. Only used when gson is NULL.
verbose	logical
nPerm	The number of permutations.
method	method of calculating the pvalue, one of "multilevel", "monte carlo" and "fgsea"
adaptive	logical, whether to use adaptive method for calculating pvalue
minPerm	minimal number of permutations for adaptive method
maxPerm	maximal number of permutations for adaptive method
pvalThreshold	pvalue threshold for adaptive method
seed	random seed for reproducibility, set to a number (or TRUE to use a fixed default seed) to make the result reproducible, or FALSE (default) to draw a random seed on each run, so results may vary between runs. The underlying permutation engine uses its own RNG seeded with this value; see `enrichit::gsea()` for details.
...	other parameters passed to `enrichit::gsea_gson()`

Value

gseaResult object

Author(s)

Guangchuang Yu https://yulab-smu.top

---

gseGO *gseGO*

Description

Gene Set Enrichment Analysis of Gene Ontology

Usage

```
gseGO(
  geneList,
  ont = "BP",
  OrgDb,
  keyType = "ENTREZID",
  exponent = 1,
  minGSSize = 10,
  maxGSSize = 500,
  eps = 1e-10,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  verbose = TRUE,
  nPerm = 1000,
  method = "multilevel",
  adaptive = FALSE,
  minPerm = 101,
  maxPerm = 1e+05,
  pvalThreshold = 0.1,
  seed = FALSE,
  ...
)
```

Arguments

geneList	order ranked geneList
ont	one of "BP", "MF", and "CC" subontologies, or "ALL" for all three.
OrgDb	OrgDb
keyType	keytype of gene
exponent	weight of each step
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
eps	boundary for calculating the p value in multilevel mode

pvalueCutoff	pvalue Cutoff
pAdjustMethod	pvalue adjustment method
verbose	print message or not
nPerm	The number of permutations.
method	method of calculating the pvalue, one of "multilevel", "permute" and "sample"
adaptive	logical, whether to use adaptive method for calculating pvalue
minPerm	minimal number of permutations for adaptive method
maxPerm	maximal number of permutations for adaptive method
pvalThreshold	pvalue threshold for adaptive method
seed	random seed for reproducibility, set to a number (or TRUE to use a fixed default seed) to make the result reproducible, or FALSE (default) to draw a random seed on each run, so results may vary between runs. The underlying permutation engine uses its own RNG seeded with this value; see `enrichit::gsea()` for details.
...	other parameters passed to `enrichit::gsea_gson()`

Value

gseaResult object

Author(s)

Yu Guangchuang

`gseKEGG` *gseKEGG*

Description

Gene Set Enrichment Analysis of KEGG

Usage

```
gseKEGG(
  geneList,
  organism = "hsa",
  keyType = "kegg",
  exponent = 1,
  minGSSize = 10,
  maxGSSize = 500,
  eps = 1e-10,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  verbose = TRUE,
```

```
  use_internal_data = FALSE,
  nPerm = 1000,
  method = "multilevel",
  adaptive = FALSE,
  minPerm = 101,
  maxPerm = 1e+05,
  pvalThreshold = 0.1,
  seed = FALSE,
  ...
)
```

Arguments

geneList	order ranked geneList
organism	supported organism listed in 'https://www.genome.jp/kegg/catalog/org_list.html'; alternatively a 'GSON' object (e.g. from 'gson_KEGG()') holding the annotation, which allows the analysis to run without contacting KEGG
keyType	one of "kegg", 'ncbi-geneid', 'ncib-proteinid' and 'uniprot'
exponent	weight of each step
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
eps	boundary for calculating the p value in multilevel mode
pvalueCutoff	pvalue Cutoff
pAdjustMethod	pvalue adjustment method
verbose	print message or not
use_internal_data	logical, use KEGG.db or latest online KEGG data
nPerm	The number of permutations.
method	method of calculating the pvalue, one of "multilevel", "permute" and "sample"
adaptive	logical, whether to use adaptive method for calculating pvalue
minPerm	minimal number of permutations for adaptive method
maxPerm	maximal number of permutations for adaptive method
pvalThreshold	pvalue threshold for adaptive method
seed	random seed for reproducibility, set to a number (or TRUE to use a fixed default seed) to make the result reproducible, or FALSE (default) to draw a random seed on each run, so results may vary between runs. The underlying permutation engine uses its own RNG seeded with this value; see `enrichit::gsea()` for details.
...	other parameters passed to `enrichit::gsea_gson()`

Value

gseaResult object

Author(s)

Yu Guangchuang

---

gseMKEGG *gseMKEGG*

Description

Gene Set Enrichment Analysis of KEGG Module

Usage

```
gseMKEGG(
  geneList,
  organism = "hsa",
  keyType = "kegg",
  exponent = 1,
  minGSSize = 10,
  maxGSSize = 500,
  eps = 1e-10,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  verbose = TRUE,
  nPerm = 1000,
  method = "multilevel",
  adaptive = FALSE,
  minPerm = 101,
  maxPerm = 1e+05,
  pvalThreshold = 0.1,
  seed = FALSE,
  ...
)
```

Arguments

geneList	order ranked geneList
organism	supported organism listed in 'https://www.genome.jp/kegg/catalog/org_list.html'; alternatively a 'GSON' object (e.g. from 'gson_KEGG()') holding the annotation, which allows the analysis to run without contacting KEGG
keyType	one of "kegg", 'ncbi-geneid', 'ncib-proteinid' and 'uniprot'
exponent	weight of each step
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
eps	boundary for calculating the p value in multilevel mode

pvalueCutoff	pvalue Cutoff
pAdjustMethod	pvalue adjustment method
verbose	print message or not
nPerm	The number of permutations.
method	method of calculating the pvalue, one of "multilevel", "permute" and "sample"
adaptive	logical, whether to use adaptive method for calculating pvalue
minPerm	minimal number of permutations for adaptive method
maxPerm	maximal number of permutations for adaptive method
pvalThreshold	pvalue threshold for adaptive method
seed	random seed for reproducibility, set to a number (or TRUE to use a fixed default seed) to make the result reproducible, or FALSE (default) to draw a random seed on each run, so results may vary between runs. The underlying permutation engine uses its own RNG seeded with this value; see `enrichit::gsea()` for details.
...	other parameters passed to `enrichit::gsea_gson()`

Value

gseaResult object

Author(s)

Yu Guangchuang

`gsePC` *gsePC*

Description

GSEA analysis for Pathway Commons

Usage

```
gsePC(geneList, eps = 1e-10, ...)
```

Arguments

geneList	a ranked gene list
eps	boundary for calculating the p value in multilevel mode
...	additional parameters, see also the parameters supported by the GSEA() function

Details

This function performs GSEA using Pathway Commons

Value

A gseaResult instance

---

gseWP *gseWP*

Description

GSEA analysis for WikiPathways

Usage

```
gseWP(geneList, organism, eps = 1e-10, ...)
```

Arguments

geneList	ranked gene list
organism	supported organisms, which can be accessed via the get_wp_organisms() function
eps	boundary for calculating the p value in multilevel mode
...	additional parameters, see also the parameters supported by the GSEA() function

Details

This function performs GSEA using WikiPathways

Value

A gseaResult instance

Author(s)

Guangchuang Yu

gson_GO *gson_KEGG*

Description

download the latest version of KEGG pathway and stored in a 'GSON' object

Usage

```
gson_GO(OrgDb, keytype = "ENTREZID", ont = "BP")
```

Arguments

OrgDb	OrgDb
keytype	keytype of genes.
ont	one of "BP", "MF", "CC", and "ALL"

Value

a 'GSON' object

gson_GO_local *Build a gson object that annotate Gene Ontology*

Description

Build a gson object that annotate Gene Ontology

Usage

```
gson_GO_local(data, ont = c("ALL", "BP", "CC", "MF"), species = NULL, ...)
```

Arguments

data	a two-column data.frame of original GO annotation. The columns are "gene_id" and "go_id".
ont	type of GO annotation, which is "ALL", "BP", "MF", or "CC". default: "ALL".
species	name of species. Default: NULL.
...	pass to ‘gson::gson()‘ constructor.

Value

a ‘gson‘ instance

Examples

```
data = data.frame(gene_id = "gene1",
            go_id = c("GO:0035492", "GO:0009764", "GO:0031040", "GO:0033714", "GO:0036349"))
gson_GO_local(data, species = "E. coli")
```

`gson_KEGG` *gson_KEGG*

Description

download the latest version of KEGG pathway and stored in a 'GSON' object

Usage

```
gson_KEGG(species, KEGG_Type = "KEGG", keyType = "kegg")
```

Arguments

`species`	species
`KEGG_Type`	one of "KEGG" and "MKEGG"
`keyType`	one of "kegg", 'ncbi-geneid', 'ncib-proteinid', 'uniprot', and 'ko'.

Value

a 'GSON' object

Author(s)

Guangchuang Yu

`gson_KEGG_mapper` *Build KEGG annotation for novel species using KEGG Mapper*

Description

KEGG Mapper service can annotate protein sequences for novel species with KO database, and KO annotation need to be converted into Pathway or Module annotation, which can then be used in 'clusterProfiler'

Usage

```
gson_KEGG_mapper(
  file,
  format = c("BLAST", "Ghost", "Kofam"),
  type = c("pathway", "module"),
  species = NULL,
  ...
)
```

Arguments

file	the name of the file which comes from the KEGG Mapper service, see Details for file format
format	string indicate format of KEGG Mapper result
type	string indicate annotation database
species	your species, NULL if ignored
...	pass to gson::gson()

Details

File is a two-column dataset with K numbers in the second column, optionally preceded by the user's identifiers in the first column. This is consistent with the output files of automatic annotation servers, BlastKOALA, GhostKOALA, and KofamKOALA. KOALA (KEGG Orthology And Links Annotation) is KEGG's internal annotation tool for K number assignment of KEGG GENES using SSEARCH computation. BlastKOALA and GhostKOALA assign K numbers to the user's sequence data by BLAST and GHOSTX searches, respectively, against a nonredundant set of KEGG GENES. KofamKOALA is a new member of the KOALA family available at GenomeNet using the HMM profile search, rather than the sequence similarity search, for K number assignment. see https://www.kegg.jp/blastkoala/, https://www.kegg.jp/ghostkoala/ and https://www.genome.jp/tools/kofamkoala/ for more information.

Value

a gson instance

Examples

```
## Not run:
 file = system.file('extdata', "kegg_mapper_blast.txt", package='clusterProfiler')
 gson_KEGG_mapper(file, format = "BLAST", type = "pathway")

## End(Not run)
```

---

gson_KO *gson_KO*

Description

Download KEGG Orthology annotation as a GSON object

Usage

```
gson_KO()
```

Value

a 'GSON' object

gson_WP *gson_WP*

Description

Download the latest version of WikiPathways data and stored in a 'GSON' object

Usage

```
gson_WP(organism)
```

Arguments

organism	supported organism, which can be accessed via the get_wp_organisms() function.

idType *idType*

Description

list ID types supported by annoDb

Usage

```
idType(OrgDb = "org.Hs.eg.db")
```

Arguments

OrgDb	annotation db

Value

character vector

Author(s)

Guangchuang Yu

`interpret` *Interpret Enrichment Results Using LLMs*

Description

Functions for interpreting functional enrichment analysis results using Large Language Models. Supports single-call interpretation, multi-agent deep analysis, and hierarchical cluster strategies.

Built on top of aisdk's 'generate_object()' for reliable structured output, and the Agent/Session system for multi-agent pipelines.

Sends enrichment results along with optional experimental context to an LLM to generate a structured biological interpretation, hypothesis, and narrative suitable for a publication.

Usage

```
interpret(
  x,
  context = NULL,
  n_pathways = NULL,
  model = NULL,
  add_ppi = FALSE,
  gene_fold_change = NULL,
  gene_stats = NULL,
  prior = NULL,
  max_tokens = 8192,
  temperature = 0.3,
  reduce = TRUE,
  reduce_threshold = NULL,
  reasoning = NULL,
  verbose = FALSE,
  reduce_go = NULL,
  go_reduction_threshold = NULL
)
```

Arguments

`x`	An enrichment result object (`enrichResult`, `gseaResult`, `compareClusterResult`, or a `data.frame` with pathway columns).
`context`	A string describing the experimental background (e.g., "scRNA-seq of mouse myocardial infarction at day 3").
`n_pathways`	Maximum number of terms (per cluster) sent to the model, taken in order of adjusted p-value after reduction. `NULL` (default) sends every term that passes the object's cutoffs; with `reduce = TRUE`, those left after `reduce_redundancy()`. On five public lists a cap below the number of representatives named fewer themes of the full result; for a model with a short context, lower `reduce_threshold` (for example 0.3) before setting a cap.

`model` Optional LLM model. When `NULL` (default), uses the aisdk package-wide default model configured via `aisdk::set_model()`. You can also supply a model ID in `provider:model` format (e.g., `"deepseek:deepseek-chat"`, `"gemini:gemini-2.5-flash"`) or a `LanguageModelV1` object. Bare model names are supported with a warning (e.g., `"deepseek-chat"`).

`add_ppi` Logical, whether to query STRING PPI network data. Default FALSE.

`gene_fold_change` Named numeric vector of log fold changes for expression context. For a `gseaResult` left `NULL`, the ranked list the GSEA was run on (`x@geneList`) is used, so the leading-edge genes reach the model with their values.

`gene_stats` Optional data frame of per-gene statistics from the differential analysis behind `x` (a limma `topTable()`, DESeq2 `results()`, edgeR `topTags()` or Seurat `FindMarkers()` table). Recognised columns: the gene (`gene`, `SYMBOL`, `symbol`, `gene_symbol`, `ENTREZID`, `ID`, or the row names), the log fold change (`log2FC`, `logFC`, `log2FoldChange`, `avg_log2FC`) and the adjusted p-value (`padj`, `adj.P.Val`, `FDR`, `p_val_adj`, `p.adjust`). When supplied, the genes of the enriched terms are listed with log fold change and FDR, and the most significant up- and down-regulated genes are added, so that the genes the model names are read from the data rather than from the term names. It also supplies `gene_fold_change` when that is `NULL`.

`prior` Optional results of other tools to be checked against the enrichment evidence (a SingleR or scGPT label, a TF-activity ranking): a string, or a named list or vector with one element per cluster. They are sent as claims to check, not as evidence, and the result gains `conclusion` (the answer to the question the context poses), `prior_verdict` (Confirmed, Refined, Corrected or Not assessable) and `prior_rationale`. Left `NULL` (default), the request is the one made before this argument existed.

`max_tokens` Maximum tokens for the LLM response. Default 8192. Some models (especially reasoning models) may need much higher values (e.g., 16384 or more) to produce complete structured output.

`temperature` Sampling temperature. Default 0.3.

`reduce` Logical. Run `reduce_redundancy()` before building the prompt, so every significant theme reaches the model once: Wang similarity for GO results, Jaccard overlap of member genes for KEGG and other gene-set collections. Default `TRUE`; a no-op for data frames and lists.

`reduce_threshold` Similarity threshold passed to `reduce_redundancy()`. `NULL` (default) uses its default of 0.7.

`reasoning` Optional reasoning/thinking configuration passed through to the provider, e.g. `list(type = "enabled", budget_tokens = 4096)`. Left `NULL` the request is byte-identical to one made before this argument existed. Whatever is passed is recorded in the execution stamp as `reasoning_requested`, alongside `reasoning_evidence` – whether the response actually carried a reasoning block.

`verbose` Logical, whether to print debug messages showing raw API responses, token usage, and JSON parsing details. Default FALSE. Equivalent to setting `options(aisdk.debug = TRUE)` for the call.

reduce_go, go_reduction_threshold
: Earlier names of reduce and reduce_threshold, still accepted; a call that passes reduce_go keeps its earlier default threshold of 0.7.

Details

Uses generate_object() internally for reliable structured output with automatic JSON repair, eliminating manual parsing failures.

Value

An interpretation object (list): overview, regulatory_drivers, key_mechanisms, crosstalk, hypothesis, narrative, refined_network and network_evidence; with prior, also conclusion, prior_verdict and prior_rationale.

Examples

```
## Not run:
# Basic usage with a data frame
df <- data.frame(
  ID = c("GO:0006915", "GO:0008284"),
  Description = c("apoptotic process", "positive regulation of proliferation"),
  GeneRatio = c("10/100", "20/100"),
  p.adjust = c(0.01, 0.02),
  geneID = c("TP53/BAX", "MYC/CCND1/CDK4")
)
res <- interpret(df,
  model = "deepseek:deepseek-chat",
  context = "Cancer proliferation study"
)
# Every significant term left after reduce_redundancy() (the default)
ego <- enrichGO(gene = genes, OrgDb = org.Mm.eg.db, ont = "BP")
res <- interpret(ego)
# A shorter prompt for a model with a short context: merge more terms per theme
res <- interpret(ego, reduce_threshold = 0.3)
# Reuse aisdk's global default model
# aisdk::set_model("openai:gpt-4o-mini")
# res <- interpret(df, context = "Cancer proliferation study")
print(res)

## End(Not run)
```

---

interpret-evidence *Interpret an evidence bundle*

Description

Hand an [evidence()] bundle to the interpretation pipeline, restricted to the entities the evidence actually supports. This is the same call as ‘interpret()‘ on an enrichment result – ‘interpret()‘ dispatches on its input – so there is no second function to learn.

Arguments

x	An ‘evidenceBundle‘ from [evidence()].
...	Further arguments passed to the interpretation pipeline, such as ‘context‘, ‘model‘ or ‘max_tokens‘.

Details

The restriction is the point. Only entries that ‘adjudicate()‘ grades ‘supported‘ or ‘single_source‘ **and** that carry level ‘A‘ or ‘B‘ reach the prompt. A level C finding is a correlation: letting it into the prompt is how a narrative ends up asserting a mechanism nobody demonstrated. Everything excluded is still reported, in ‘$evidence_excluded‘, so the omission is visible rather than silent.

‘unsupported_claims‘ afterwards re-reads the model’s prose and lists the entities it named that were not in ‘$evidence_admitted‘. It reuses [evidence_status()] on each collection rather than on the bundle, because ‘evidence_status()‘ expects an object with a ‘geneID‘ / ‘core_enrichment‘ column and a bundle has neither.

Value

An ‘interpretation‘ object with two extra elements: ‘evidence_admitted‘ (the rows that reached the prompt) and ‘evidence_excluded‘ (the rows that did not, with the reason).

See Also

[evidence()], [adjudicate()], [interpret()]

Examples

```
## Not run:
ev <- evidence(c("HIF1A", "PGK1", "CA9"))
interpret(ev)

## End(Not run)
```

---

interpret_agent *Interpret enrichment results using a multi-agent pipeline (Deep Mode)*

Description

Employs specialized AI agents in sequence for rigorous interpretation, and is the entry point that combines the enrichment evidence with the results of other tools (`prior`) into a conclusion:

1. Agent Cleaner: Filters noise and selects relevant pathways.
2. Agent Detective: Identifies key regulators and functional modules.
3. Agent Synthesizer: Produces a coherent biological narrative.
4. Agent Reviewer (optional, `review = TRUE`): Audits the draft for grounding, overclaiming, and experimental-label fidelity; flags issues and returns a corrected overview.

Usage

```
interpret_agent(
  x,
  context = NULL,
  prior = NULL,
  n_pathways = NULL,
  model = NULL,
  add_ppi = FALSE,
  gene_fold_change = NULL,
  max_tokens = 8192,
  temperature = 0.3,
  review = FALSE,
  reduce = TRUE,
  reduce_threshold = NULL,
  reasoning = NULL,
  verbose = FALSE,
  reduce_go = NULL,
  go_reduction_threshold = NULL
)
```

Arguments

x	An enrichment result object.
context	A string describing the experimental background. State the question the analysis should answer here (for example "which cell type is this cluster?"); the answer is returned as `conclusion`.
prior	Optional results of other tools to be checked against the enrichment evidence, as text: a SingleR or scGPT label, a ranked list of transcription factors from a regulon analysis, the protein complexes enriched in the gene list, and so on. A single string applies to every cluster; a named character vector or list supplies one entry per cluster. The Detective and the Synthesizer see it as a claim to check, not as evidence, and the result gains `prior_verdict` (Confirmed, Refined, Corrected or Not assessable) and `prior_rationale`.
n_pathways	Maximum number of terms (per cluster) the Cleaner starts from, taken in order of adjusted p-value after reduction. `NULL` (default) uses every term that passes the object's cutoffs; with `reduce = TRUE`, those left after `reduce_redundancy()`.
model	Optional LLM model. When `NULL` (default), uses the aisdk package-wide default model configured via `aisdk::set_model()`. You can also supply a model ID in `provider:model` format or a `LanguageModelV1` object. Bare model names are supported with a warning.
add_ppi	Logical, whether to query PPI data. Default FALSE.
gene_fold_change	Named numeric vector of log fold changes. When supplied, every input gene is also listed to the Detective and the Synthesizer, most up-regulated first.
max_tokens	Maximum tokens per agent call. Default 8192.
temperature	Sampling temperature. Default 0.3.

review — Logical, whether to run a final Reviewer agent (Agent 4) that audits the synthesized draft against the input evidence for grounding, overclaiming, experimental-label fidelity, and **evidence sufficiency** (whether the supplied evidence classes can support a claim as specific as the draft makes). When TRUE, the result gains a review field (verdict, grounding_score, evidence_sufficiency, supported_resolution, abstain_recommended, issues, adjusted_confidence, warnings) and overview_original; if the verdict is "revise", overview is replaced by the corrected version, in which claims exceeding supported_resolution are downgraded to that resolution. Default FALSE, which reproduces the released three-agent behavior exactly: the audit is an addition a caller opts into, and leaving it off keeps existing scripts measuring what they measured before. Note that the audit roughly doubles wall-clock time per unit and adds one model call, and that it is not infallible – on objects where the Synthesizer returned nothing it has been observed to return pass with a High grounding score.

reduce — Logical. Run reduce_redundancy() before building any prompt, as interpret() does: Wang similarity for GO results, Jaccard overlap of member genes for other gene-set collections. Deterministic and makes no model call. Default TRUE.

reduce_threshold — Similarity threshold passed to reduce_redundancy(). NULL (default) uses its default of 0.7.

reasoning — Optional reasoning/thinking configuration passed through to the provider, e.g. list(type = "enabled", budget_tokens = 4096). Left NULL the request is byte-identical to one made before this argument existed. Whatever is passed is recorded in the execution stamp as reasoning_requested, alongside reasoning_evidence – whether the response actually carried a reasoning block. The two are separate because a gateway can accept the parameter and drop it, and the caller can only observe the second.

verbose — Logical, whether to print debug messages. Default FALSE.

reduce_go, go_reduction_threshold — Earlier names of reduce and reduce_threshold, still accepted; a call that passes reduce_go keeps its earlier default threshold of 0.7.

Details

Uses aisdk's Agent and Session system for shared context across agents.

Value

An interpretation object with deep analysis fields, conclusion (the answer to the question the context poses), plus regulatory_drivers, refined_network, and network_evidence from the detective agent. When prior is supplied, also prior_verdict and prior_rationale. When review = TRUE, also includes review and overview_original. The review field reports grounding separately from resolution: abstain_recommended = "yes" flags a draft whose cited evidence is real but which resolves to a more specific entity than that evidence can single out, with supported_resolution giving the claim the evidence does support.

Examples

```
## Not run:
res <- interpret_agent(df,
  model = "openai:gpt-4o",
  context = "scRNA-seq of mouse MI day 3"
)
print(res)
# Check another tool's answer against the enrichment evidence
res <- interpret_agent(ego_cluster3,
  context = "PBMC scRNA-seq; which cell type is cluster 3?",
  gene_fold_change = fc3, prior = "SingleR (HPCA): T_cell:CD4+_Naive"
)
res$conclusion; res$prior_verdict

## End(Not run)
```

`interpret_dialogue` *Discuss an evidence bundle with a model*

Description

Open a conversation about an [evidence()] result. Unlike [interpret()], which produces one finished narrative, this returns a *dialogue object* the caller keeps and sends further questions to, so a mechanism hypothesis can be argued with rather than announced.

Usage

```
interpret_dialogue(
  x,
  context = NULL,
  model = NULL,
  assertions = character(0),
  max_turns = 50L,
  species = NULL
)

## S3 method for class 'evidenceDialogue'
print(x, ...)
```

Arguments

`x`	An ‘evidenceDialogue‘.
`context`	Optional free text describing the experiment.
`model`	A model specification understood by ‘aisdk‘, e.g. ‘"deepseek:deepseek-v4-flash"‘.
`assertions`	Character vector of things the caller knows that the data does not, e.g. ‘"our CEBPB knockdown showed no phenotype"‘. Recorded as level ‘U‘ and never used to grade anything.

max_turns	Stop accepting questions after this many. Counted in questions, not transcript rows: a question writes two rows and an assertion one, so counting rows made the limit mean different things depending on how the caller used the dialogue.
species	Species whose annotation bounds the check, e.g. ‘"Homo sapiens"‘. Taken from the bundle when ‘NULL‘. Without a matching OrgDb the dialogue refuses to open rather than running a check that would pass everything.
...	Ignored.

Details

The conversation is bounded by what the evidence says, and the boundary is enforced rather than requested. Three rules hold for every turn:

* Only entities that [adjudicate()] already reports may be discussed. A question about a regulator the evidence does not contain is answered with that fact, not with model knowledge, because a plausible answer about an unexamined factor is indistinguishable from an invented one. * Anything the caller adds – a lab observation, an unpublished result – is recorded as a **user assertion** with level ‘U‘. It enters the conversation and the transcript, and it never changes a verdict: ‘CEBPB‘ stays ‘single_source‘ even after the user says their knockdown had no phenotype. The alternative would let a sentence in the chat raise the grade of a finding, which is exactly the claim the grades exist to protect. * Every reply is checked against the admit list, so a model that names an unexamined regulator is caught the same way [interpret()] catches it.

Value

An ‘evidenceDialogue‘: a list with ‘session‘ (the live ‘aisdk::ChatSession‘), ‘transcript‘ (a data frame of turns), ‘assertions‘, and ‘admitted‘ (the entities a reply may name). Print it to see the conversation; call [ask()] to continue it.

The session is a reference object and is mutated in place, so assigning the result of [ask()] rebinds the variable but does not copy the conversation. Use [fork_dialogue()] to take a branch that can be abandoned.

‘x‘, invisibly.

See Also

[ask()], [evidence()], [adjudicate()]

Examples

```
## Not run:
ev <- evidence(genes, upstream = c("tf", "perturbagen"),
               design = list(treatment = "knockdown"))
d  <- interpret_dialogue(ev, context = "airway smooth muscle, dex vs untreated")
d  <- ask(d, "Why is CEBPB ranked first?")
d  <- say(d, "Our CEBPB knockdown showed no phenotype in this line.")

## End(Not run)
```

`interpret_hierarchical` *Interpret enrichment results using a hierarchical strategy*

Description

First interprets major clusters to establish lineage context, then interprets sub-clusters with hierarchical constraints from the major cluster annotations.

Usage

```
interpret_hierarchical(
  x_minor,
  x_major,
  mapping,
  model = NULL,
  max_tokens = 8192,
  temperature = 0.3,
  n_pathways = NULL,
  reduce = TRUE,
  reduce_threshold = NULL
)
```

Arguments

`x_minor`	Enrichment result for sub-clusters.
`x_major`	Enrichment result for major clusters.
`mapping`	A named vector mapping sub-cluster IDs to major cluster IDs.
`model`	Optional LLM model. When 'NULL' (default), uses the aisdk package-wide default model configured via 'aisdk::set_model()'. You can also supply a model ID in 'provider:model' format or a 'LanguageModelV1' object. Bare model names are supported with a warning.
`max_tokens`	Maximum tokens. Default 8192.
`temperature`	Sampling temperature. Default 0.3.
`n_pathways, reduce, reduce_threshold`	Passed to both levels, as in `interpret()`: by default every significant term left after [reduce_redundancy()].

Value

An 'interpretation_list' object with a cell-type label ('cell_type', 'confidence', 'reasoning', 'markers') per sub-cluster.

ko2name *ko2name*

Description

convert ko ID to descriptive name

Usage

```
ko2name(ko)
```

Arguments

ko	ko ID

Value

data.frame

Author(s)

guangchuang yu

---

mechanism_alias_regulator *Bring a regulator label to one spelling*

Description

The collections and the perturbation databases do not agree on symbol spelling: C3:TFT names the hypoxia factor 'HIF1' (the legacy symbol) while GPSAdb names it 'HIF1A', and the collections say 'GR' where the database says 'NR3C1'. Without this, one factor supported by both its target sets and a knockdown arrives as two findings and can never be corroborated.

Usage

```
mechanism_alias_regulator(regulator)
```

Arguments

regulator	Character vector of regulator labels.

Details

Exported because a caller comparing the package's labels against an outside database hits the same mismatch, and a hand-written alias table drifts from this one.

Value

The same vector, with aliases replaced by their canonical symbol.

See Also

[evidence()], [adjudicate()]

Examples

```
mechanism_alias_regulator(c("HIF1", "GR", "HIF1A"))
```

`mechanism_discover` *Rank candidate transcription factors for a differential expression profile*

Description

This is the cheap first stage. It ranks transcription factors by correlating each one's signed target weights against the whole differential expression vector, then leaves the decision to [pathway_evidence()], which checks a named candidate against measured perturbation data.

Usage

```
mechanism_discover(
  signed,
  network,
  top = 10L,
  min_targets = 5L,
  rank_by = c("score", "abs"),
  keep_self_targets = FALSE
)
```

Arguments

`signed`	Named numeric vector of log2 fold changes, names are gene symbols. The whole vector is used, not just the significant genes: the correlation is what separates a factor's targets from everything else.
`network`	Data frame of transcription factor targets with columns 'source' (factor), 'target' (gene symbol) and optionally 'weight' (positive for activation targets, negative for repression targets; 1 when absent). CollecTRI and DoRothEA can be exported in this shape. Identical repeated edges are deduplicated; conflicting weights for the same source-target pair are rejected rather than resolved by row order.
`top`	Integer. How many candidates to return.
`min_targets`	Integer. Factors with fewer measured targets than this are skipped: a correlation over three genes is noise.

`rank_by` Character. ‘"score"‘ (default) ranks by the signed score, so factors that look active come first; that is the right default for treatment and ligand experiments, where the driver is switched on. ‘"abs"‘ ranks by magnitude regardless of sign, which is what a knockdown or knockout profile needs: its driver scores negative and would otherwise sink to the bottom. On our benchmark 16 of 17 profiles are activation types, so ‘"score"‘ wins there; that is a property of the corpus, not of the method.

`keep_self_targets` Logical. When ‘FALSE‘ (the default) a factor that lists itself as a target loses that edge. Self-targets inflate the score for any knockdown or knockout, because the factor’s own gene is the strongest signal in the profile; in CollecTRI 62 reach evaluation carry such an edge, and removing them costs about five percentage points of ranking performance.

Details

The score is the t-value of that correlation. **A positive score means the factor looks active in this experiment, a negative one means repressed.** The ranking is not a verdict. In a frozen exploratory corpus of 17 profiles, all 17 expected drivers were present and scorable with CollecTRI. Using the default signed ranking, expected-label hit counts were 6/17 at rank one, 10/17 in the top five and 13/17 in the top ten. The corpus includes one knockout and repeated studies/drivers; these are not independent validation estimates or proof of unique causal identification. A rank outside the first twenty must not be counted as network absence. Treat the output as a shortlist.

In the truncated top-twenty candidate pool, score and measured target count were positively associated, including among non-reference candidates (Spearman correlation about 0.24). This selected-pool observation does not establish a general size bias or show whether annotation depth, correlated targets or biology explains it. ‘n_targets‘ and ‘share‘ disclose coverage. No target-count weight is applied: exploratory adjustments have not been independently validated.

Value

A data frame with one row per candidate: ‘rank‘, ‘source‘, ‘score‘ (the t-value), ‘r‘ (the correlation), ‘direction‘ (‘"activates"‘ or ‘"represses"‘), ‘n_targets‘, and ‘share‘ (the fraction of all measured genes that are targets of this factor; the denominator is ‘length(signed)‘, not the number of genes in the network).

See Also

[pathway_evidence()] to test the shortlist against perturbation data, and [mechanism_preflight()] to decide whether the profile can carry any claim at all.

Examples

```
set.seed(2)
genes <- paste0("g", 1:50)
signed <- stats::setNames(c(rep(1.5, 5), stats::rnorm(45, 0, 0.2)), genes)
network <- data.frame(source = "TF1", target = genes[1:5], weight = 1)
mechanism_discover(signed, network)
```

`mechanism_graph` *The mechanism the evidence permits, as a mechgraph*

Description

Turns an ‘evidenceBundle‘ into a ‘mechgraph‘ object: candidate regulators on one side, the input genes they are credited with on the other, every node and edge carrying the grade it earned. Draw it with ‘plot()‘ (rendered by ggtangle) or keep it as data.

Usage

```
mechanism_graph(x, max_genes = 8, species = NULL, release = NULL)
```

Arguments

`x`	An ‘evidenceBundle‘ from [evidence()].
`max_genes`	Most genes to keep per regulator, by overlap size. A collection term can list a thousand genes and the figure stops being readable long before that.
`species`	Species whose collections supply the membership.
`release`	MSigDB release, taken from the bundle when ‘NULL‘.

Details

A ‘mechgraph‘ rather than a bare data frame, because the object already carries the contract this analysis needs: a node has an ‘id‘, a ‘type‘ and a ‘label‘; an edge has a ‘from‘, a ‘to‘ and a ‘type‘. Anything else rides along as a column and is passed through to igraph, so the grade travels with the node instead of being re-joined at plot time.

A link means **"this collection lists this gene"**, not "this factor regulates this gene". The collection is a target repertoire, and which of its members are real targets is exactly what the analysis could not settle. That distinction is why an edge here is evidence of association regardless of the regulator’s own grade.

Absent and untested entities are not included. They are the larger part of most results and including them would bury the findings; [adjudicate()] lists them.

Value

A ‘mechgraph‘ object. ‘nodes‘ carry ‘id‘, ‘type‘ (‘"regulator"‘, ‘"family"‘ or ‘"gene"‘), ‘label‘, ‘grade‘ and ‘verdict‘; ‘edges‘ carry ‘from‘, ‘to‘, ‘type‘ (‘"target"‘), ‘term‘, ‘level‘ and ‘source‘. ‘metadata‘ records the input size and the species.

See Also

[plot_mechanism()], [evidence()], [adjudicate()]

Examples

```
## Not run:
ev <- evidence(genes, upstream = c("tf", "perturbagen"),
               design = list(treatment = "knockdown"))
mg <- mechanism_graph(ev)
plot(mg)

## End(Not run)
```

`mechanism_preflight` *Screen input completeness and reference coverage before driver comparison*

Description

Run this before [pathway_evidence()]. It returns a heuristic tier, not evidence of causation: ‘attribution‘ means the checked input and coverage floors pass; ‘description‘ means a floor failed or a required check is unknown; ‘none‘ means there are no pathways to describe.

Usage

```
mechanism_preflight(
  x,
  signed = NULL,
  reference = NULL,
  drivers = NULL,
  min_up = 50L,
  min_members = 5L,
  min_cell_lines = 2L,
  min_background = 10L
)

## S3 method for class 'mechanismPreflight'
print(x, ...)
```

Arguments

`x`	An enrichment result, or a data.frame of pathways.
`signed`	Named numeric of log fold changes for every gene measured. If neither these measurements nor the enrichment input gene count are available, the gene-count check is unknown and the tier is ‘description‘.
`reference`	Optional ‘mechanismReference‘; used to count candidate cell lines and background programmes. Without it, candidate coverage is queried from the inventory.

drivers	Optional candidate drivers. At least one must meet the coverage floor for 'attribution'. Omitted candidates or failed inventory lookups leave coverage unknown and cap the tier at 'description'. A driver absent from a supplied reference has known zero coverage in that reference.
min_up	Up-regulated genes required before attribution is offered.
min_members	Median pathway size required. This is the pathway's own size, not how many of its members the reference happens to cover: the 12-member floor recorded in the design belongs to the perturbation test's p-value, not to this descriptive input screen.
min_cell_lines	Cell lines a driver needs before attribution is offered.
min_background	Comparable reference programmes required for the specificity percentile to mean anything.
...	Ignored.

Details

The boundaries are the ones already written down in the pathway-mechanism design (differential genes, pathway size, driver coverage), not thresholds fitted to a benchmark. They are still crude: the design notes record that 23 up-regulated genes are clearly too few and 351 clearly enough, so the cut points below sit between two measured points rather than on a measured one. The numbers behind the tier are returned with it, so a caller can move the cut points and see which datasets change tier.

Value

A 'mechanismPreflight' list with 'tier', 'reasons', 'metrics' and 'counts'.

See Also

[pathway_evidence()], [mechanism_report()]

---

mechanism_report *Report candidate consistency, conflicts, and evidence gaps*

Description

Summarise measured evidence without treating a nonsignificant test as causal exclusion. A supported row is consistency with a directional explanation, not identification of a unique upstream driver. Significant conflicting tests remain visible even when other datasets support that candidate.

Usage

```
mechanism_report(
  x,
  drivers = NULL,
  inventory = NULL,
  context = list(),
```

```
  preflight = NULL,
  network = NULL,
  min_overlap = 0.1,
  global_effect_de = 3000L
)

## S3 method for class 'mechanismReport'
print(x, n = 6L, ...)
```

Arguments

x	A table from [pathway_evidence()]. Its ‘perturbation_tests‘ attribute retains all candidate/dataset outcomes, not just the best result per pathway.
drivers	Drivers to inventory. Defaults to the drivers requested in ‘x‘ and those named in its rows or retained tests.
inventory	Optional available-dataset table with ‘perturbed_gene‘ and optionally ‘dataset_id‘, ‘cell_line‘, ‘tissue‘, ‘n_up‘ and ‘n_down‘. Without it GPSAdb is queried (up to 100 rows per resolved spelling). A failed lookup is unknown coverage, not evidence that no datasets exist. A supplied table describes only its own scope, not an exhaustive database inventory.
context	Named list. ‘context$tissue‘ is compared with available tissues.
preflight	Optional [mechanism_preflight()] result. If the input screen does not support attribution, additional experiments are not automatically ranked; the report explains the current limit without claiming that future experiments could never change the evidence.
network	Optional regulator-target table with ‘source‘ and ‘target‘. ‘module‘ is an unsigned pairwise Jaccard neighbourhood among the requested candidates. It ignores edge weights and is not a disjoint clustering, proof of biological equivalence, or evidence that expression experiments cannot distinguish its members. No module accuracy gain is assumed.
min_overlap	Jaccard threshold in [0, 1], default ‘0.10‘. Lower thresholds enlarge neighbourhoods and their chance coverage; this is descriptive.
global_effect_de	Median DE gene count above which perturbation breadth is flagged, default ‘3000‘. Counts depend on tissue, method, sample size and DE thresholds. The flag does not prove essentiality or pathway nonspecificity, and does not change a grade or score.
n	Number of rows to show per section.
...	Ignored.

Value

A ‘mechanismReport‘ list with ‘verdicts‘ (classes ‘supported‘, ‘conflicting‘, ‘undetermined‘), ‘gaps‘, ‘next_steps‘, ‘summary‘, ‘context‘, ‘inventory‘, ‘perturbation_tests‘ and ‘candidate_evidence‘. The last two expose all retained tests and counts of supporting/conflicting results per pathway/candidate. ‘next_steps$unlocks‘ is retained for compatibility and counts rows addressed by an action, not

promised grade upgrades. The legacy ‘coherent-refuted‘ verdict is treated as undetermined, not disproved. ‘verdicts$evidence_status‘ is the primary presentation: ‘support‘, ‘conflict‘, ‘mixed‘, ‘undetermined‘ or ‘noncomputable‘, derived from all retained tests per pathway, not from its grade. ‘availability‘, ‘incomplete‘ and the attempted/computable, nonsignificant, noncomputable and failed counts disclose gaps. Missing ledgers do not establish support. Legacy ‘level‘, ‘class‘ and ‘verdict‘ remain secondary compatibility fields. Ledger support does not imply that member support passed its floor or establish causal identity.

See Also

[pathway_evidence()], [mechanism_preflight()]

`mechanism_reproduction` *Concordance of a pathway shift across independent perturbation datasets*

Description

For each (driver, pathway) pair, takes the mean log2 fold change of the pathway members in every available perturbation dataset of that driver, and reports how many of them move the pathway the same way. The majority sign is the **mode of the signs**, not the sign of the sum: one dataset with a very large effect must not overturn two smaller ones that agree with each other. Datasets with no measured members are dropped before counting, so ‘k‘ is the number that actually contributed, not the number declared in ‘meta‘.

Usage

```
mechanism_reproduction(
  profiles,
  meta,
  pathways,
  drivers = NULL,
  min_datasets = 2L,
  min_biosamples = 2L,
  min_genes = 3L,
  max_log2fc = NULL
)
```

Arguments

`profiles`	Numeric matrix, rows are genes (symbols), columns are perturbation dataset identifiers.
`meta`	Data frame with columns ‘dataset‘ and ‘driver‘. Optional ‘biosample‘ (for the independent-sample count) and ‘logFC‘ (the knockdown strength, used by ‘max_log2fc‘).
`pathways`	Named list: pathway name to a character vector of member genes. [pathway_evidence()] builds this from the enrichment result.

drivers	Character vector of drivers to score. Defaults to every driver present in ‘meta‘.
min_datasets	Integer. Fewer datasets than this cannot show concordance (one dataset is trivially concordant), so the pair is reported as ‘not-computable‘.
min_biosamples	Integer. A row only counts as reproduced when the supporting datasets come from at least this many distinct known biosamples. Missing or blank biosamples do not establish independent replication.
min_genes	Integer. Datasets with fewer finite pathway members than this are excluded before counting ‘k‘ or computing concordance. A pair with fewer than ‘min_datasets‘ remaining is reported as ‘not-computable‘.
max_log2fc	Optional numeric. When given and ‘meta‘ has a ‘logFC‘ column, datasets weaker than this knockdown are dropped before scoring.

Details

This is the input [pathway_evidence()] needs to decide how strongly a single-cell-line level-B row may be worded.

Value

A data frame with one row per (driver, pathway): ‘driver‘, ‘pathway‘, ‘k‘ (datasets with usable measurements), ‘n_biosamples‘, ‘n_genes‘ (members found in ‘profiles‘), ‘concordance‘, ‘majority_sign‘, and ‘grade‘ which is one of ‘reproduces‘, ‘consistent‘, ‘not-reproduced‘ or ‘not-computable‘.

Examples

```
set.seed(1)
profiles <- matrix(rnorm(60), nrow = 20,
                   dimnames = list(paste0("g", 1:20), c("d1", "d2", "d3")))
profiles[1:5, ] <- -1
meta <- data.frame(dataset = c("d1", "d2", "d3"),
                   driver = "TF1", biosample = c("A", "B", "B"))
mechanism_reproduction(profiles, meta, list(pw = paste0("g", 1:5)))
```

---

mechanism_sets *Upstream gene-set collections for mechanism analysis*

Description

Download an MSigDB collection whose members are genes and whose set identifiers name an *upstream participant* (a transcription factor, a microRNA, a perturbagen, a cell state or a tissue) rather than a biological process. Feeding such a collection to [enricher()] or [GSEA()] turns a gene list into a rank of candidate mechanisms, which is the opposite direction from a pathway analysis.

Usage

```
mechanism_sets(
  source = c("tf", "tfunion", "mirna", "perturbagen", "cellstate", "tissue"),
  species = "Homo sapiens",
  release = "2026.2.Hs",
  cache_dir = getOption("clusterProfiler_cache_dir"),
  subset = NULL
)
```

Arguments

source	One of ‘"tf"‘ (C3:TFT), ‘"tfunion"‘ (C3:TFT plus four public transcription-factor collections, see [mechanism_tfunion()]), ‘"mirna"‘ (C3:MIR), ‘"perturbagen"‘ (C2:CGP), ‘"cellstate"‘ (C8) or ‘"tissue"‘ (C4:CGN).
species	Currently only human (‘"Homo sapiens"‘) is supported, because the files addressed here are the ‘*.Hs.*‘ release.
release	MSigDB release string, used verbatim in the download URL and recorded in the ‘version‘ slot. Pinning it as a literal makes the collection contents reproducible; a release is never inferred from the current date.
cache_dir	Directory holding downloaded GMT files. Defaults to ‘getOption("clusterProfiler_cache_dir")‘, which [clusterProfiler-package] sets once per session.
subset	Keep only the parts of the collection produced by a named evidence rule: ‘"gtrd"‘ for the ChIP-defined TF sets (whose identifiers are gene symbols), ‘"motif"‘ for the predicted ones, ‘"unnamed_motif"‘ for consensus sequences that name no factor. Measured on the airway list, ‘subset = "gtrd"‘ takes C3:TFT’s share of family-or-consensus labels from 40 ChIP target sets overlap this list less than chance. Read before setting it.

Details

The returned object is a [gson][gson::gson]-style ‘GSON‘, accepted by ‘enricher(..., gson =)‘ and ‘GSEA(..., gson =)‘. It must go through the ‘gson =‘ argument, not ‘TERM2GENE =‘: only the ‘gson =‘ path carries ‘species‘, ‘keytype‘ and ‘gsname‘ into the result object, and [evidence()] relies on those three slots.

Value

A ‘GSON‘ object whose ‘gsid2gene‘ maps set identifiers to HGNC symbols, with ‘species‘, ‘keytype = "SYMBOL"‘, ‘gsname‘, ‘version‘ and ‘accessed_date‘ filled in.

See Also

[evidence()] for the analysis that consumes these collections. ‘get_ppi_network()‘ for the network-side evidence source.

Examples

```
## Not run:
tf <- mechanism_sets("tf")
x <- enricher(c("HIF1A", "PGK1", "CA9"), gson = tf)

## End(Not run)
```

`mechanism_significant` *Which rows of a regulator table are significant*

Description

One rule for the whole table. A row carrying ‘padj_within_source‘ is judged by it; a row without one falls back to the source’s own ‘significant‘ flag. Neither alone is enough: a p-value of 0.6 is not made significant by a flag, and a source that reports no test (a similarity rank, say) still knows which of its rows are hits.

Usage

```
mechanism_significant(r, alpha = 0.05)
```

Arguments

`r`	A regulator ‘data.frame‘, e.g. ‘evidence(x)$regulators‘.
`alpha`	The threshold for ‘padj_within_source‘.

Details

Exported because the alternative – testing ‘< alpha‘ by hand – silently ignores the flag and treats a ‘NA‘ p-value as "not significant" rather than as "not tested".

Value

A logical vector, one per row.

See Also

[evidence()], [adjudicate()]

Examples

```
## Not run:
r <- evidence(genes, upstream = "tf")$regulators
r[mechanism_significant(r), ]

## End(Not run)
```

`mechanism_tfunion` *The transcription-factor union collection*

Description

Builds one collection from C3:TFT plus four public transcription-factor libraries, so the search for an upstream regulator is not bounded by the ~592 factors C3:TFT happens to name.

Usage

```
mechanism_tfunion(
  species = "Homo sapiens",
  release = "2026.2.Hs",
  cache_dir = getOption("clusterProfiler_cache_dir")
)
```

Arguments

`species`	Currently only human (‘"Homo sapiens"‘) is supported, because the files addressed here are the ‘*.Hs.*‘ release.
`release`	MSigDB release string, used verbatim in the download URL and recorded in the ‘version‘ slot. Pinning it as a literal makes the collection contents reproducible; a release is never inferred from the current date.
`cache_dir`	Directory holding downloaded GMT files. Defaults to ‘getOption("clusterProfiler_cache_dir")‘, which [clusterProfiler-package] sets once per session.

Details

Why this exists. C3:TFT has no target set at all for ‘EPAS1‘, ‘NFKB1‘, ‘TP53‘, ‘CTCF‘, ‘ESR1‘, ‘NOTCH1‘, ‘JUN‘ or ‘FOS‘. On a hypoxia dataset whose published answer is HIF-2 (‘EPAS1‘), no amount of grading can find a regulator the collection does not name – the ceiling was the data, not the method. These four libraries together add 315 factors C3:TFT lacks.

Why the sets stay separate. Merging each factor’s sets into one gene list would be wrong, and measurably so: CTCF’s 108 ENCODE sets union to 21295 genes, 11 set keeps its identity and is tagged with its library prefix (‘ENCODE:‘, ‘ChEA:‘, ‘TFP:‘, ‘TRRUST:‘, ‘C3TFT:‘); folding duplicates is [adjudicate()]’s job, and it has the evidence levels to do that with.

What it costs. 2869 sets against C3:TFT’s 1114, so the multiple-testing correction is spread over roughly three times as many hypotheses. Measured on the hypoxia contrast, the true signal survived that comfortably – ‘HIF1A‘ moved from padj 1.2e-4 under C3:TFT to 2.5e-27 as a ChEA set – while three random gene sets of the same size returned **zero** significant sets, so the extra hypotheses did not buy extra false positives.

Building it reads 2.68 million rows, about 9 seconds. The result is kept for the rest of the session and reused, keyed on each source file’s path, mtime and size, so replacing or re-downloading a library invalidates the entry rather than serving the old collection.

Value

A ‘GSON‘, in the same shape as [mechanism_sets()] returns.

See Also

[mechanism_sets()]

Examples

```
## Not run:
ev <- evidence(genes, upstream = "tfunion")

## End(Not run)
```

`mechanism_writable` *The sentence an evidence level permits*

Description

Turn a verdict and a level into the strongest statement the evidence supports, with the regulator name filled in. This is what keeps a level C finding out of a mechanism sentence: the wording available at C describes a correlation, and no higher wording is offered.

Usage

```
mechanism_writable(verdict, level, regulator, sources = NULL, family = NULL)
```

Arguments

`verdict`	A ‘adjudicate()‘ verdict.
`level`	An evidence level (‘"A"‘, ‘"B"‘, ‘"C"‘) or ‘NA‘.
`regulator`	The entity name.
`sources`	Comma-separated source labels as ‘adjudicate()‘ records them. A measured perturbation is graded A whether it hit a regulator or a downstream effector, so the sentence for a single-source A finding depends on whether a target-set collection also names the gene as a regulator.
`family`	Logical vector from ‘adjudicate()‘, TRUE where the label names a protein family or a complex rather than one gene. Such a label is a real answer, so it gets its own sentence: the collection names the group, not one gene, and no perturbation dataset can exist for it.

Value

A character vector of sentences.

See Also

[adjudicate()]

Examples

```
mechanism_writable(c("supported", "single_source", "untested"),
                   c("B", "C", NA), c("FOXO3", "MIR23C", "NFE2L2"))
```

`mechanism_xref` *Evidence level rules for upstream gene sets*

Description

The rules [evidence()] applies when it turns an enriched collection into a candidate mechanism. Each row is a rule, evaluated in order, and the first match wins – so the order is part of the definition, not a display choice.

Usage

```
mechanism_xref()
```

Details

The levels express how strong the claim is, and therefore how it may be written:

* **A** – the collection came from a measured perturbation. Only perturbagen evidence reaches this level, and only when ‘design$treatment‘ reports one. * **B** – a mechanism is recorded: a ChIP-supported target set, a database annotation, or a membership in a curated complex. * **C** – an association only: a predicted target repertoire, a co-expression pattern, a cell-state or tissue signature. These may be reported as correlation and never as regulation.

Value

A ‘data.frame‘ with columns ‘source‘, ‘rule‘, ‘match‘, ‘default_level‘ and ‘note‘.

See Also

[evidence()]

Examples

```
mechanism_xref()
```

merge_result *merge_result*

Description

merge a list of enrichResult objects to compareClusterResult

Usage

```
merge_result(enrichResultList)
```

Arguments

enrichResultList	a list of enrichResult objects

Value

a compareClusterResult instance

Author(s)

Guangchuang Yu

mnseGO *mnseGO*

Description

Multi-layer Network-based Gene Set Enrichment Analysis of Gene Ontology

Usage

```
mnseGO(
  seed_list,
  networks,
  couplings,
  ont = "BP",
  OrgDb,
  keyType = "ENTREZID",
  mode = c("evidence", "signed"),
  layer_weights = NULL,
  collapse = c("weighted_mean", "sum", "mean", "max_abs"),
  target_layer = NULL,
  output_space = c("union", "gene"),
  p = 0.5,
```

```
  interlayer_strength = 1,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  verbose = TRUE,
  ...
)
```

Arguments

seed_list	named list of named numeric vectors, one per layer
networks	named list of layer-specific networks
couplings	data.frame of inter-layer edges
ont	one of "BP", "MF", and "CC" subontologies, or "ALL" for all three.
OrgDb	OrgDb
keyType	keytype of gene
mode	propagation mode, one of "evidence" and "signed"
layer_weights	optional named numeric vector of layer weights
collapse	one of "weighted_mean", "sum", "mean", or "max_abs"
target_layer	optional layer name to export scores from
output_space	one of "union" and "gene"
p	restart probability for RWR
interlayer_strength	global scaling factor for coupling edges
specific_weight	logical, whether to apply gene specificity weighting
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
threshold	convergence threshold for RWR
maxIter	maximal number of RWR iterations
verbose	print message or not
...	other parameters passed to `enrichit::mnsea_gson()`

Value

mnseaResult object

mnseKEGG *mnseKEGG*

Description

Multi-layer Network-based Gene Set Enrichment Analysis of KEGG

Usage

```
mnseKEGG(
  seed_list,
  networks,
  couplings,
  organism = "hsa",
  keyType = "kegg",
  mode = c("evidence", "signed"),
  layer_weights = NULL,
  collapse = c("weighted_mean", "sum", "mean", "max_abs"),
  target_layer = NULL,
  output_space = c("union", "gene"),
  p = 0.5,
  interlayer_strength = 1,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  verbose = TRUE,
  use_internal_data = FALSE,
  ...
)
```

Arguments

seed_list	named list of named numeric vectors, one per layer
networks	named list of layer-specific networks
couplings	data.frame of inter-layer edges
organism	supported organism listed in 'https://www.genome.jp/kegg/catalog/org_list.html', or a GSON object
keyType	one of "kegg", 'ncbi-geneid', 'ncbi-proteinid' and 'uniprot'
mode	propagation mode, one of "evidence" and "signed"
layer_weights	optional named numeric vector of layer weights
collapse	one of "weighted_mean", "sum", "mean", or "max_abs"
target_layer	optional layer name to export scores from

output_space	one of "union" and "gene"
p	restart probability for RWR
interlayer_strength	global scaling factor for coupling edges
specific_weight	logical, whether to apply gene specificity weighting
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
threshold	convergence threshold for RWR
maxIter	maximal number of RWR iterations
verbose	print message or not
use_internal_data	logical, use KEGG.db or latest online KEGG data
...	other parameters passed to enrichit::mnsea_gson()

Value

mnseaResult object

---

mnseMKEGG *mnseMKEGG*

Description

Multi-layer Network-based Gene Set Enrichment Analysis of KEGG Module

Usage

```
mnseMKEGG(
  seed_list,
  networks,
  couplings,
  organism = "hsa",
  keyType = "kegg",
  mode = c("evidence", "signed"),
  layer_weights = NULL,
  collapse = c("weighted_mean", "sum", "mean", "max_abs"),
  target_layer = NULL,
  output_space = c("union", "gene"),
  p = 0.5,
  interlayer_strength = 1,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
```

```
  threshold = 1e-09,
  maxIter = 100,
  verbose = TRUE,
  ...
)
```

Arguments

seed_list	named list of named numeric vectors, one per layer
networks	named list of layer-specific networks
couplings	data.frame of inter-layer edges
organism	supported organism listed in 'https://www.genome.jp/kegg/catalog/org_list.html', or a GSON object
keyType	one of "kegg", 'ncbi-geneid', 'ncbi-proteinid' and 'uniprot'
mode	propagation mode, one of "evidence" and "signed"
layer_weights	optional named numeric vector of layer weights
collapse	one of "weighted_mean", "sum", "mean", or "max_abs"
target_layer	optional layer name to export scores from
output_space	one of "union" and "gene"
p	restart probability for RWR
interlayer_strength	global scaling factor for coupling edges
specific_weight	logical, whether to apply gene specificity weighting
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
threshold	convergence threshold for RWR
maxIter	maximal number of RWR iterations
verbose	print message or not
...	other parameters passed to `enrichit::mnsea_gson()`

Value

mnseaResult object

mnseWP *mnseWP*

Description

Multi-layer Network-based Gene Set Enrichment Analysis of WikiPathways

Usage

```
mnseWP(
  seed_list,
  networks,
  couplings,
  organism,
  mode = c("evidence", "signed"),
  layer_weights = NULL,
  collapse = c("weighted_mean", "sum", "mean", "max_abs"),
  target_layer = NULL,
  output_space = c("union", "gene"),
  p = 0.5,
  interlayer_strength = 1,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  verbose = TRUE,
  ...
)
```

Arguments

seed_list	named list of named numeric vectors, one per layer
networks	named list of layer-specific networks
couplings	data.frame of inter-layer edges
organism	supported organisms listed by `get_wp_organisms()`, or a GSON object
mode	propagation mode, one of "evidence" and "signed"
layer_weights	optional named numeric vector of layer weights
collapse	one of "weighted_mean", "sum", "mean", or "max_abs"
target_layer	optional layer name to export scores from
output_space	one of "union" and "gene"
p	restart probability for RWR
interlayer_strength	global scaling factor for coupling edges

specific_weight	logical, whether to apply gene specificity weighting
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
threshold	convergence threshold for RWR
maxIter	maximal number of RWR iterations
verbose	print message or not
...	other parameters passed to `enrichit::mnsea_gson()`

Value

mnseaResult object

`nseGO` *nseGO*

Description

Network-based Gene Set Enrichment Analysis of Gene Ontology

Usage

```
nseGO(
  geneList,
  network,
  ont = "BP",
  OrgDb,
  keyType = "ENTREZID",
  mode = c("evidence", "signed"),
  p = 0.5,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  verbose = TRUE,
  ...
)
```

Arguments

geneList	order ranked geneList
network	edge list (data.frame with 2 or 3 columns) or sparse matrix
ont	one of "BP", "MF", and "CC" subontologies, or "ALL" for all three.
OrgDb	OrgDb

keyType	keytype of gene
mode	propagation mode, one of "evidence" and "signed"
p	restart probability for RWR
specific_weight	logical, whether to apply gene specificity weighting
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
threshold	convergence threshold for RWR
maxIter	maximal number of RWR iterations
verbose	print message or not
...	other parameters passed to `enrichit::nsea_gson()`

Value

nseaResult object

`nseKEGG` *nseKEGG*

Description

Network-based Gene Set Enrichment Analysis of KEGG

Usage

```
nseKEGG(
  geneList,
  network,
  organism = "hsa",
  keyType = "kegg",
  mode = c("evidence", "signed"),
  p = 0.5,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  verbose = TRUE,
  use_internal_data = FALSE,
  ...
)
```

Arguments

geneList	order ranked geneList
network	edge list (data.frame with 2 or 3 columns) or sparse matrix
organism	supported organism listed in 'https://www.genome.jp/kegg/catalog/org_list.html', or a GSON object
keyType	one of "kegg", 'ncbi-geneid', 'ncbi-proteinid' and 'uniprot'
mode	propagation mode, one of "evidence" and "signed"
p	restart probability for RWR
specific_weight	logical, whether to apply gene specificity weighting
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
threshold	convergence threshold for RWR
maxIter	maximal number of RWR iterations
verbose	print message or not
use_internal_data	logical, use KEGG.db or latest online KEGG data
...	other parameters passed to `enrichit::nsea_gson()`

Value

nseaResult object

`nseMKEGG` *nseMKEGG*

Description

Network-based Gene Set Enrichment Analysis of KEGG Module

Usage

```
nseMKEGG(
  geneList,
  network,
  organism = "hsa",
  keyType = "kegg",
  mode = c("evidence", "signed"),
  p = 0.5,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
```

```
  maxIter = 100,
  verbose = TRUE,
  ...
)
```

Arguments

geneList	order ranked geneList
network	edge list (data.frame with 2 or 3 columns) or sparse matrix
organism	supported organism listed in 'https://www.genome.jp/kegg/catalog/org_list.html', or a GSON object
keyType	one of "kegg", 'ncbi-geneid', 'ncbi-proteinid' and 'uniprot'
mode	propagation mode, one of "evidence" and "signed"
p	restart probability for RWR
specific_weight	logical, whether to apply gene specificity weighting
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
threshold	convergence threshold for RWR
maxIter	maximal number of RWR iterations
verbose	print message or not
...	other parameters passed to `enrichit::nsea_gson()`

Value

nseaResult object

`nseWP` *nseWP*

Description

Network-based Gene Set Enrichment Analysis of WikiPathways

Usage

```
nseWP(
  geneList,
  network,
  organism,
  mode = c("evidence", "signed"),
  p = 0.5,
  specific_weight = FALSE,
  minGSSize = 10,
```

```
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  verbose = TRUE,
  ...
)
```

Arguments

geneList	order ranked geneList
network	edge list (data.frame with 2 or 3 columns) or sparse matrix
organism	supported organisms listed by `get_wp_organisms()`, or a GSON object
mode	propagation mode, one of "evidence" and "signed"
p	restart probability for RWR
specific_weight	logical, whether to apply gene specificity weighting
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of genes annotated for testing
threshold	convergence threshold for RWR
maxIter	maximal number of RWR iterations
verbose	print message or not
...	other parameters passed to `enrichit::nsea_gson()`

Value

nseaResult object

---

pathway_evidence *Test enriched pathways against measured perturbation data*

Description

Test enriched pathways against measured perturbation data

Usage

```
pathway_evidence(
  x,
  drivers = NULL,
  signed = NULL,
  alpha = 0.05,
  min_support = 0.5,
  max_pathways = 20L,
  max_datasets_per_driver = 6L,
```

```
  reproduction = NULL
)

## S3 method for class 'pathway_evidence'
print(x, n = 10L, ...)
```

Arguments

x	An ‘enrichResult‘ or ‘gseaResult‘ from [enrichGO()], [enrichKEGG()] and friends, or a ‘$pathways‘ table from an [evidence()] bundle.
drivers	Character vector of candidate driver genes. Each is resolved to its perturbation datasets and tested against every pathway.
signed	Optional named numeric of log2 fold changes, used to work out which way a pathway’s members moved when the result does not say.
alpha	Significance threshold for a perturbation to count.
min_support	Minimum fraction of members moving the same way before a pathway counts as moved at all.
max_pathways	How many pathways to report, taken in the result’s own order. This also caps the run: every pathway is tested against every driver dataset over the network, so the wall time grows with ‘max_pathways * drivers * datasets‘. Measured on the hypoxia pilot, 25 pathways took 96 calls and about a minute, 60 took 210 calls and about three. Results are memoised by (dataset, member set), which helps when pathways share members and not otherwise, so pick this deliberately rather than leaving it high.
max_datasets_per_driver	Cap on how many datasets are tested per driver. Testing all of them is ‘n_drivers * n_datasets‘ calls and grows without bound; six is enough to see two cell lines agree, which is all level A needs. Datasets from distinct cell lines are preferred over extra replicates of one line, because replication is what the grade turns on. The cap applies to datasets tested, not to the lookups that resolve them: a candidate is searched under one or more removal spellings, so an unknown symbol can cost a handful of search calls before it is reported as empty.
reproduction	Optional data frame from [mechanism_reproduction()]. When given, a level-B row whose only support is a single cell line is worded according to how many independent perturbation datasets agree: a pair that reproduces says so, a pair that does not reproduce is labelled unverified, and a pair with no second dataset remains consistent but under-evidenced.
n	For ‘print()‘, how many rows to show.
...	Ignored, for method consistency.

Value

A ‘data.frame‘, one row per pathway, classed ‘pathway_evidence‘ so it prints as a short summary rather than twenty columns. Two columns carry the interpretation: ‘statement‘ holds a claim a perturbation experiment could refute, and ‘reason‘ says why a row stopped short of level A (‘single-cell-line‘, ‘direction-conflict‘, ‘no-dataset‘, ‘too-small‘, ‘not-significant‘, ‘member-derived‘ and the

input-side cases). Level A leaves ‘reason‘ as ‘NA‘. Level A is weaker than its name suggests: a perturbation that moves the pathway also moves it if the perturbation is unrelated but perturbs transcription broadly, and four of thirty background drivers reached A on the hypoxia pilot. Read A as consistency worth following up, not as identification of the driver; the statement says as much. A level-B row distinguishes ‘coherent-untested‘ (no test could run) from the legacy key ‘coherent-refuted‘ (tests ran but none reached significance; this is not evidence of falsity). When the answer cannot reach A for a structural reason the call also warns. The ‘perturbation_tests‘ attribute is a data frame with character columns ‘pathway‘, ‘driver‘, ‘dataset‘, ‘cell_line‘, ‘status‘, ‘direction‘ and double columns ‘es‘, ‘padj‘. It retains each candidate/dataset outcome, including ‘supports‘ (significant, opposite the pathway direction), ‘conflicts‘ (significant, same direction), ‘not-significant‘, ‘failed‘, and ‘not-computable‘ (missing direction, missing/nonfinite statistics or zero ES). These statuses describe consistency, not unique causal identification. A known member support at or above ‘min_support‘ is required for promotion based on significant perturbation evidence. Replication is counted within the selected candidate only; contradictory datasets remain in the attribute. The ‘dataset_lookup‘ attribute records, per requested driver, whether the lookup returned data (‘available‘), succeeded but found nothing usable (‘empty‘), or could not be performed (‘failed‘, with the error text). A failed lookup yields reason ‘lookup-failed‘ and is never reported as the database holding no data; ‘not-run‘ marks an empty input that never reached dataset resolution. With several drivers, a partial failure is disclosed in that attribute and by a warning rather than in the row ‘reason‘, which keeps describing the candidate that was actually tested.

`plot.interpretation` *plot*

Description

plot

Usage

```
## S3 method for class 'interpretation'
plot(x, layout = "nicely", ...)
```

Arguments

`x`	An ‘interpretation‘ object.
`layout`	Graph layout, default is "nicely".
`...`	Additional arguments passed to ‘ggplot2::ggplot‘.

plot_mechanism *Draw a mechanism graph*

Description

Renders the ‘mechgraph‘ from [mechanism_graph()] with ggtangle, colouring nodes by the grade they earned.

Usage

```
plot_mechanism(x, layout = "nicely", ...)
```

Arguments

x	A ‘mechgraph‘ from [mechanism_graph()].
layout	Layout passed to [ggtangle::ggplot.mechgraph]. ‘"nicely"‘ suits a few findings; ‘"kk"‘ spreads a crowded one.
...	Passed to [ggtangle::ggplot.mechgraph].

Value

A ‘ggplot‘ object.

See Also

[mechanism_graph()]

Examples

```
## Not run:
plot_mechanism(mechanism_graph(ev))

## End(Not run)
```

plot_mechanism_tree *Draw the mechanism as a tree*

Description

Renders the attribution from [attribute_genes()] with ggtree: one branch per regulator, its assigned genes as tips, coloured by the grade that won the assignment.

Renders the attribution from [attribute_genes()] with ggtree: one branch per regulator, its assigned genes as tips, coloured by the grade that won the assignment.

Usage

```
plot_mechanism_tree(
  x,
 layout = c("tree_and_leaf", "dendrogram", "rectangular", "slanted", "fan", "circular"),
  ...
)

plot_mechanism_tree(
  x,
 layout = c("tree_and_leaf", "dendrogram", "rectangular", "slanted", "fan", "circular"),
  ...
)
```

Arguments

x	A ‘mechgraph‘ from [mechanism_graph()], or the data frame from [attribute_genes()].
layout	Passed to [ggtree::ggtree()]. The default ‘"tree_and_leaf"‘ puts the regulators in an inner ring and their genes outside, which keeps the two sets of labels apart; ‘"rectangular"‘ reads top to bottom and suits a few regulators; ‘"fan"‘ and ‘"dendrogram"‘ are also available.
...	Passed to [ggtree::ggtree()].

Details

A tree rather than a network because the question a mechanism has to answer is directional – which factor accounts for which genes – and a network draws every claim with the same weight, leaving the reader to resolve overlaps by eye. The tree states the resolution: when three regulators list the same gene, one of them is named.

The tree is drawn as a fan when there are many regulators and as a rectangle otherwise, because a fan with three branches is mostly empty space.

A tree rather than a network because the question a mechanism has to answer is directional – which factor accounts for which genes – and a network draws every claim with the same weight, leaving the reader to resolve overlaps by eye. The tree states the resolution: when three regulators list the same gene, one of them is named.

The tree is drawn as a fan when there are many regulators and as a rectangle otherwise, because a fan with three branches is mostly empty space.

Value

A ‘ggtree‘ plot.

A ‘ggtree‘ plot.

See Also

[attribute_genes()], [plot_mechanism()]

Examples

```
## Not run:
plot_mechanism_tree(mechanism_graph(ev))

## End(Not run)
## Not run:
plot_mechanism_tree(mechanism_graph(ev))

## End(Not run)
```

---

plotGOgraph *plotGOgraph*

Description

plot GO graph

Usage

```
plotGOgraph(
  x,
  firstSigNodes = 10,
  useInfo = "all",
  sigForAll = TRUE,
  useFullNames = TRUE,
  ...
)
```

Arguments

x	output of enrichGO or gseGO
firstSigNodes	number of significant nodes (retangle nodes in the graph)
useInfo	additional info
sigForAll	if TRUE the score/p-value of all nodes in the DAG is shown, otherwise only score will be shown
useFullNames	logical
...	additional parameter of showSigOfNodes, please refer to topGO

Value

GO DAG graph

Author(s)

Guangchuang Yu

print.adjudication *Print an adjudication*

Description

Shows what a reader needs in order to decide whether to go on to [interpret()]: how many findings reached each verdict, and which regulators the strongest ones name. The per-entity table is not printed – at a few hundred entities it buries the summary it is supposed to support.

Usage

```
## S3 method for class 'adjudication'
print(x, ...)
```

Arguments

x	An ‘adjudication‘ from [adjudicate()].
...	Ignored.

Value

‘x‘, invisibly.

print.evidenceBundle *Print an evidence bundle*

Description

Show the input size, the collections used and the most significant regulators, one row per regulator with its evidence level. The cutoffs come from ‘print_pvalue‘ / ‘print_qvalue‘ recorded when the bundle was built; they affect this display only, never the table stored in ‘x$regulators‘.

Usage

```
## S3 method for class 'evidenceBundle'
print(x, ...)
```

Arguments

x	An ‘evidenceBundle‘ from [evidence()].
...	Ignored.

Value

‘x‘, invisibly.

provenance *Execution provenance of an interpretation*

Description

Returns the execution record attached to a result of `interpret()`, `interpret_agent()` or `interpret_hierarchical()`: the model actually addressed, the decoding parameters requested (and whether the response confirmed them), SHA-256 digests of the system prompt, user prompt, raw response and input evidence, token usage, `finish_reason`, per-call status and package versions.

Usage

```
provenance(x)
```

Arguments

`x`	An `interpretation`, `interpretation_list` or agent result.

Details

A remote model is not a versionable dependency. Without this record an interpretation cannot be reproduced, and a call that failed at the transport layer is indistinguishable from a model that had nothing to say – both arrive as an ordinary result object. Check `provenance(x)$ok` before treating any interpretation as data.

Value

An `interpret_provenance` list, or `NULL` if the object carries no stamp.

read.gmt.pc *read.gmt.pc*

Description

Parse gmt file from Pathway Common

Usage

```
read.gmt.pc(gmtfile, output = "data.frame")
```

Arguments

`gmtfile`	A gmt file
`output`	one of 'data.frame' or 'GSON'

Details

This function parse gmt file downloaded from Pathway common

Value

A data.frame or A GSON object depends on the value of 'output'

`reduce_redundancy` *Reduce redundant terms in an enrichment result before interpretation*

Description

Removes near-duplicate significant terms with `simplify()`, so that a long result can be shown to the model without repeating its near-synonyms. GO results are compared by Wang semantic similarity over the GO DAG (GOSemSim); every other gene-set collection (KEGG, Reactome, WikiPathways, MSigDB, custom `TERM2GENE`) by the Jaccard overlap of the member genes the result reports (`geneID`, or `core_enrichment` for GSEA). For each term, of the terms more similar to it than `threshold`, the one with the smallest adjusted p-value is kept and the others are removed. A lower `threshold` removes more. Only terms that pass the object's cutoffs are considered; a `compareClusterResult` is reduced within each cluster, and a GO result spanning several ontologies within each ontology. Deterministic, and makes no model call. `interpret()` and `interpret_agent()` call it by default (`reduce = TRUE`).

Usage

```
reduce_redundancy(
  x,
  threshold = NULL,
  method = c("auto", "Wang", "jaccard"),
  semData = NULL
)
```

Arguments

`x`	An `enrichResult`, `gseaResult` or `compareClusterResult`.
`threshold`	Similarity above which the less significant of two terms is removed. `NULL` (default) uses 0.7, the default of `simplify()`, for both measures.
`method`	`"auto"` (default) uses `"Wang"` when every term is a GO term and `"jaccard"` otherwise; either can be forced. Wang on non-GO terms falls back to Jaccard.
`semData`	Optional `GOSemSimDATA` for the Wang measure; built with `GOSemSim::godata(ont = )` when `NULL`. Wang reads only the GO DAG, so no OrgDb is needed.

Value

`x` with the redundant significant terms removed. Attributes record the step: `reduced`, `n_original`, `n_reduced`, `reduction_method` and `reduction_threshold`, and the legacy `rrvgo_*` names read by earlier analysis scripts.

See Also

simplify(), which does the actual reduction.

Examples

```
## Not run:
ego <- enrichGO(gene = genes, OrgDb = org.Hs.eg.db, ont = "BP")
reduce_redundancy(ego)                # Wang, 0.7
kk <- enrichKEGG(gene = genes, organism = "hsa")
reduce_redundancy(kk)                 # Jaccard on member genes, 0.7

## End(Not run)
```

---

say *Tell an evidence dialogue something the data does not contain*

Description

Records a caller-supplied fact – an unpublished observation, a lab result, a constraint – so later answers can take it into account.

Usage

```
say(dialogue, statement, ...)
```

Arguments

dialogue	An ‘evidenceDialogue‘.
statement	One thing the caller knows, in their own words.
...	Ignored.

Details

The fact is stored as a **user assertion** at level ‘U‘ and is deliberately inert with respect to grading. It cannot raise a verdict, because a sentence typed into a chat is not a measurement, and letting one move ‘CEBPB‘ from ‘single_source‘ to ‘supported‘ would make the grade meaningless.

Value

The updated ‘evidenceDialogue‘, with the assertion appended and sent to the model as context for later turns.

See Also

[interpret_dialogue()], [ask()]

search_kegg_organism *search_kegg_organism*

Description

search kegg organism, listed in https://www.genome.jp/kegg/catalog/org_list.html

Usage

```
search_kegg_organism(
  str,
  by = "scientific_name",
  ignore.case = FALSE,
  use_internal_data = TRUE
)
```

Arguments

str	string
by	one of 'kegg.code', 'scientific_name' and 'common_name'
ignore.case	TRUE or FALSE
use_internal_data	logical, use kegg_species.rda or latest online KEGG data

Value

data.frame

Author(s)

Guangchuang Yu

simplify *simplify method*

Description

simplify output from enrichGO and gseGO by removing redundancy of enriched GO terms

simplify output from compareCluster by removing redundancy of enriched GO terms

Usage

```
## S4 method for signature 'enrichResult'
simplify(
  x,
  cutoff = 0.7,
  by = "p.adjust",
  select_fun = min,
  measure = "Wang",
  semData = NULL
)

## S4 method for signature 'gseaResult'
simplify(
  x,
  cutoff = 0.7,
  by = "p.adjust",
  select_fun = min,
  measure = "Wang",
  semData = NULL
)

## S4 method for signature 'compareClusterResult'
simplify(
  x,
  cutoff = 0.7,
  by = "p.adjust",
  select_fun = min,
  measure = "Wang",
  semData = NULL
)
```

Arguments

x	output of enrichGO
cutoff	similarity cutoff
by	feature to select representative term, selected by 'select_fun' function
select_fun	function to select feature passed by 'by' parameter
measure	method to measure similarity. One of '"Wang"' (default), '"Resnik"', '"Rel"', '"Jiang"', '"Lin"', '"TC"' for GO semantic similarity, or '"JC"' / '"overlap"' to compare the term gene sets, which works for every collection (KEGG, Reactome, MSigDB, ...)
semData	GOSemSimDATA object

Details

'simplify()' removes redundant enriched terms by grouping the terms whose similarity exceeds 'cutoff' and keeping one representative per group.

For GO results the similarity is semantic and comes from GOSemSim (‘measure = "Wang"‘, ‘"Resnik"‘, ‘"Rel"‘, ‘"Jiang"‘, ‘"Lin"‘, ‘"TC"‘), which needs an ontology and a ‘GOSemSimDATA‘ object.

For KEGG, Reactome, MSigDB and any other gene-set collection, set ‘measure = "JC"‘ (Jaccard index of the gene sets) or ‘measure = "overlap"‘ (shared genes over the smaller set). These measures are computed from ‘@geneSets‘, need no ontology, and therefore work on any enrichment result, including Reactome and KEGG results for which GO semantic similarity is undefined (#743). A Reactome or KEGG hierarchy gives many significantly enriched terms that share most of their genes; keeping the best-scoring term of each overlapping group compresses them into the distinct biological programs.

Value

updated enrichResult object

updated compareClusterResult object

Author(s)

Guangchuang Yu

Gwang-Jin Kim and Guangchuang Yu

References

issue #28 https://github.com/GuangchuangYu/clusterProfiler/issues/28

issue #162 https://github.com/GuangchuangYu/clusterProfiler/issues/162

See Also

[enrichplot::pairwise_termsim]

`uniprot_get` *uniprot_get*

Description

retreve annotation data from uniprot

Usage

```
uniprot_get(taxID)
```

Arguments

`taxID`	taxonomy ID

Value

gene table data frame

Author(s)

guangchuang yu

Index

∗ **classes**
groupGOResult-class, 41
∗ **datasets**
DataSet, 16
∗ **manip**
compareCluster, 13
enrichGO, 19
enrichKEGG, 21
groupGO, 39

add_go_level, 4
adjudicate, 5
append_kegg_category, 7
ask, 7
attribute_genes, 8

bitr, 9
bitr_kegg, 10
browseKEGG, 11

chat, 11
collapse_findings, 12
compareCluster, 13
consensus_findings, 15

DataSet, 16
DE_GSE8057 *(*DataSet*)*, 16
download_KEGG, 16
dropGO, 17

enrichDAVID, 17
enricher, 18
enrichGO, 19
enrichKEGG, 21
enrichMKEGG, 22
enrichPC, 23
enrichWP, 24
evidence, 25
evidence_biotrigger, 28
evidence_perturbation, 29
evidence_ppi, 31
evidence_status, 32

fork_dialogue, 33

gcSample *(*DataSet*)*, 16
get_ppi *(*getPPI*)*, 35
get_ppi_network, 33
get_wp_organisms, 34
getPPI, 35
getTaxID, 36
getTaxInfo, 37
Gff2GeneTable, 37
go2ont, 38
go2term, 38
gofilter, 39
groupGO, 39
groupGOResult-class, 41
GSEA, 41
gseGO, 43
gseKEGG, 44
gseMKEGG, 46
gsePC, 47
gseWP, 48
gson_GO, 49
gson_GO_local, 49
gson_KEGG, 50
gson_KEGG_mapper, 50
gson_KO, 51
gson_WP, 52

idType, 52
interpret, 53
interpret(), *32*, *58*, *93*, *94*
interpret-evidence, 55
interpret_agent, 56
interpret_agent(), *32*, *93*, *94*
interpret_dialogue, 59
interpret_hierarchical, 61
interpret_hierarchical(), *32*, *93*

kegg_category *(*DataSet*)*, 16

kegg_species *(DataSet)*, 16
ko2name, 62

mechanism_alias_regulator, 62
mechanism_discover, 63
mechanism_graph, 65
mechanism_preflight, 66
mechanism_report, 67
mechanism_reproduction, 69
mechanism_sets, 70
mechanism_significant, 72
mechanism_tfunion, 73
mechanism_writable, 74
mechanism_xref, 75
merge_result, 76
mnseGO, 76
mnseKEGG, 78
mnseMKEGG, 79
mnseWP, 81

nseGO, 82
nseKEGG, 83
nseMKEGG, 84
nseWP, 85

pathway_evidence, 86
plot.interpretation, 88
plot_mechanism, 89
plot_mechanism_tree, 89
plotGOgraph, 91
print.adjudication, 92
print.collapsedFindings *(collapse_findings)*, 12
print.evidenceBundle, 92
print.evidenceDialogue *(interpret_dialogue)*, 59
print.mechanismPreflight *(mechanism_preflight)*, 66
print.mechanismReport *(mechanism_report)*, 67
print.pathway_evidence *(pathway_evidence)*, 86
provenance, 93

read.gmt.pc, 93
reduce_redundancy, 94
reduce_redundancy(), *53, 54*, *57, 58*

say, 95
search_kegg_organism, 96
show,groupGOResult-method *(groupGOResult-class)*, 41
simplify, 96
simplify(), *94, 95*
simplify,compareClusterResult-method *(simplify)*, 96
simplify,enrichResult-method *(simplify)*, 96
simplify,gseaResult-method *(simplify)*, 96

uniprot_get, 98