Package 'NEArender'

November 28, 2026

Title Render Omics Profiles in Functional Network Space

Version 2.0

Maintainer Andrey Alexeyenko <andrej.alekseenko@scilifelab.se>

Description Tools for transforming gene-level omics profiles into pathway-level scores using methods of network enrichment (Alexeyenko et al., 2012, <doi:10.1186/1471-2105-13-226>), random walks, over-representation, network texture (Alexeyenko, 2026 <doi:10.64898/2026.07.12.737672>), spectral graph analysis (Chung, 1997 <ISBN 978-0-8218-0315-8>), and gene-set variation methods (Hänzelmann et al., 2013 <doi:10.1186/1471-2105-14-7>), with optional enrichment adjustment for restricted gene universes (pathway enrichment of small diagnostic panels, targeted sequencing etc.). Input uses on original gene/protein vs. sample matrices; output is pathway vs. sample matrices, i.e. the sample dimension is preserved. Main downstream applications of the rendered sample space include:

1) using the pathway profiles as predictor variables, where the advantages are dimensionality reduction and predictor robustness (Franco et al. <doi:10.1038/s41598-019-39019-2>);

2) discovery of cancer drivers (Petrov and Alexeyenko, 2022 <doi:10.7554/eLife.74010>);

3) differential pathway enrichment, as a more sensitive and informative alternative differential gene expression (Alexeyenko et al., 2022 <doi:10.21037/tlcr-22-236>).

The package maintains network objects in igraph, custom, and sparse matrix formats, parallelized computations, and provides rich functionality for API access to pathway and network databases as well as gene/protein ID management and compatibility.

License GPL-3

URL https://github.com/andreyalexeyenko/NEArender

BugReports https://github.com/andreyalexeyenko/NEArender/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports curl, igraph, jsonlite, Matrix, methods, parallel

Suggests GSVA, knitr, KernSmooth, MASS, rmarkdown, testthat (>= 3.0.0), venn

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Andrey Alexeyenko [aut, cph, cre] (ORCID:
<https://orcid.org/0000-0001-8812-6481>),
Ashwini Jeggari [ctb]

Repository CRAN

Date/Publication 2026-11-28 15:00:20 UTC

Contents

NEArender-package . . . 3
as_igraph_network . . . 4
as_nearender_network . . . 5
as_sparse_network . . . 6
available_databases . . . 7
available_id_formats . . . 8
available_ontology_collections . . . 8
convert_network . . . 9
download_database . . . 10
download_geneset_database . . . 11
download_ndex_network . . . 12
download_network_database . . . 13
download_ontology_collection . . . 14
download_resource . . . 15
ear . . . 16
export_network . . . 18
filter_ontology_collection . . . 19
glcm.net . . . 20
gsva.render . . . 21
haralick.matrix . . . 22
haralick.net . . . 24
humanbase_spec . . . 25
import.net . . . 26
import_do_collection . . . 26
import_gene_sets . . . 27
import_go_collection . . . 30
import_kegg_pathways . . . 31
import_network . . . 32
load_database . . . 35
load_geneset_database . . . 37
load_id_mappings . . . 38
load_ndex_network . . . 38
load_network_database . . . 40
load_ontology_collection . . . 42
manifest . . . 43

map_identifiers . . . 44
master_id_format . . . 45
merge_networks . . . 45
mutations2ags . . . 47
ndex_network_summary . . . 48
nea.render . . . 49
NEArender.test . . . 50
nearender_result . . . 51
nea_member_recovery_benchmark . . . 52
network_catalog . . . 54
network_overlap . . . 54
network_qc . . . 55
node_degrees . . . 56
normalization_info . . . 57
ontology_gene_set_api . . . 58
ora.render . . . 65
pagerank.render . . . 66
plot_subgraph . . . 68
prepare_id_mappings . . . 69
prepare_resource . . . 70
read_network . . . 71
reduce_network . . . 73
render_panel . . . 74
roc . . . 74
run_renderer . . . 76
rwr.render . . . 78
samples2ags . . . 80
save_gs_list . . . 81
search_ndex_networks . . . 82
set.heat . . . 84
spectral.render . . . 85
spectral_eigenpairs . . . 87
string_channel_spec . . . 89
texture.render . . . 89
topology_2nd . . . 92
universal_member_recovery_benchmark . . . 93

Index **95**

`NEArender-package` *NEArender: Render omics profiles in functional network space*

Description

NEArender converts gene-level measurements and altered gene sets into pathway-by-sample scores. It provides network enrichment, over-representation, random-walk, network-texture, spectral, and GSVA-family renderers together with explicit network, gene-set, database, and identifier-mapping utilities. For restricted gene universes, such as diagnostic panels, `ear()` can empirically calibrate NEA or ORA against equally sized random sets sampled from the specified universe.

Main workflows

- Prepare altered gene sets with `samples2ags()` or `mutations2ags()`.
- Render them with `nea.render()`, `ora.render()`, `rwr.render()`, or `pagerank.render()`.
- Render continuous profiles with `texture.render()`, `spectral.render()`, or `gsva.render()`.
- Inspect remote data options with `available_databases()` and identifier mappings with `available_id_formats()`. Downloads are always explicit.
- Discover arbitrary NDEx networks with `search_ndex_networks()`, then use explicit snapshot download and offline loading.

Renderer functions return a nearender_result object.

Author(s)

Maintainer: Andrey Alexeyenko <andrej.alekseenko@scilifelab.se> (ORCID) [copyright holder]

Other contributors:

- Ashwini Jeggari [contributor]

See Also

Useful links:

- https://github.com/andreyalexeyenko/NEArender
- Report bugs at https://github.com/andreyalexeyenko/NEArender/issues

---

as_igraph_network *Convert a network to igraph*

Description

Convert legacy NEArender adjacency lists or indexed sparse networks to a simple undirected igraph object. A selected source edge score or legacy weight is stored in the canonical igraph `weight` edge attribute. The stored legacy edge count is checked against the topology.

Usage

```
as_igraph_network(network, case_sensitive = FALSE)
```

Arguments

network	An igraph, sparse matrix, legacy NEArender, or indexed NEArender network.
case_sensitive	Preserve identifier case.

Value

A simple undirected igraph object.

See Also

as_sparse_network(), as_nearender_network(), convert_network()

Examples

```
legacy <- list(links = list(a = "b", b = "a"), Ntotal = 1,
  weights = list(a = c(b = 2), b = c(a = 2)))
graph <- as_igraph_network(legacy)
igraph::edge_attr(graph, "weight")
```

---

as_nearender_network *Convert a network to the NEArender named-adjacency-list representation*

Description

Convert an igraph or indexed network to the named adjacency-list representation used by NEArender. The igraph `weight` attribute is preserved as adjacency-aligned legacy `weights`; renderers still decide explicitly whether those weights define their analysis.

Usage

```
as_nearender_network(network, case_sensitive = FALSE)
```

Arguments

network	An igraph, sparse matrix, legacy NEArender, or indexed NEArender network.
case_sensitive	Preserve identifier case.

Value

A list with `links` and `Ntotal`, and `weights` when edge weights are available.

See Also

as_igraph_network(), as_sparse_network(), convert_network()

Examples

```
graph <- import_network(data.frame(from = "A", to = "B", score = 2),
  score_column = "score", species = NULL, input_id_format = "custom",
  verbose = FALSE)
legacy <- as_nearender_network(graph)
legacy$weights
```

`as_sparse_network` *Convert a network to the NEArender sparse representation*

Description

The returned object stores binary topology in `adjacency` and, when available, edge weights in a separate `weights` sparse matrix. Keeping these matrices separate prevents source edge scores from silently becoming transition probabilities in graph-based calculations.

Usage

```
as_sparse_network(network, case_sensitive = FALSE)
```

Arguments

`network`	An igraph, sparse matrix, legacy NEArender, or indexed NEArender network.
`case_sensitive`	Preserve identifier case.

Value

A `nearender_sparse_network` object.

See Also

`as_igraph_network()`, `as_nearender_network()`, `convert_network()`

Examples

```
graph <- import_network(data.frame(from = "A", to = "B", score = 2),
  score_column = "score", species = NULL, input_id_format = "custom",
  verbose = FALSE)
sparse <- as_sparse_network(graph)
sparse$adjacency
sparse$weights
```

`available_databases` *List curated databases*

Description

Return the built-in, offline catalog of fixed-release remote resources and licensed user-supplied sources. Listing the catalog never makes a network request. Entries that aggregate source data with different reuse terms, including MSigDB collections, are marked as requiring explicit acceptance. The catalog includes collection names, plain-language collection annotations, and expected gene set counts where applicable. For Pathway Commons, `score_semantics` describes the derived edge weight: the distinct supporting-PubMed count plus a 0.5 pseudocount.

Usage

```
available_databases(
  type = c("all", "network", "pathways", "gene_sets"),
  species = NULL
)
```

Arguments

`type`	Database type to list: `"all"`, `"network"`, or `"pathways"`. `"gene_sets"` is accepted as a synonym for `"pathways"`.
`species`	Optional scientific name, common name, package species ID, three-letter KEGG style organism ID, or NCBI taxonomy ID. By default, return resources for every supported species.

Value

A data frame describing supported database releases, including resource type, organism, collection, identifiers, source edge score field and semantics, license, expected size, checksum, acquisition mode, and expected gene-set count where applicable.

See Also

`download_database()`, `load_database()`

Examples

```
human_networks <- available_databases("network", species = "9606")
human_networks[, c("id", "version", "provider", "source_score_column")]
available_databases("pathways", species = "mouse")[,
  c("id", "collection", "expected_gene_sets")]
```

available_id_formats *List supported identifier formats*

Description

Return the offline catalog of species-specific gene, transcript, and protein identifiers available for conversion. For each species, the catalog lists the master identifier and alternative formats so that input gene sets and networks can be converted to the same identifier space. When multiple source identifiers map to one master identifier, they are collapsed during conversion. Mapping tables can be prepared on demand from remote providers; listing the catalog does not make a network request.

Usage

```
available_id_formats(species = NULL)
```

Arguments

species	Optional scientific name, common name, package species ID, three-letter KEGG style organism ID, or NCBI taxonomy ID. By default, return every supported species.

Value

A data frame with one row per species and identifier format. The species column contains the three-letter KEGG style organism ID used by the package catalog, the eviNet species request field, and KEGG.

Examples

```
formats <- available_id_formats()
species <- unique(formats[, c("species_id", "species", "organism", "tax_id")])
print(species, row.names = FALSE)
available_id_formats("rat")[, c("master_id", "id_format", "category")]
```

available_ontology_collections *List release-backed ontology collections*

Description

List the Disease Ontology and organism-specific Gene Ontology products that NEArender can acquire as immutable local snapshots. This operation is local and performs no network request.

Usage

```
available_ontology_collections(ontology = c("all", "DO", "GO"), species = NULL)
```

Arguments

ontology	One of "all", "DO", or "GO".
species	Optional scientific/common name, NEArender species code, or NCBI taxon identifier supported by available_id_formats().

Value

A data frame with one row per collection and its official source products, provider, and license.

See Also

download_ontology_collection(), load_ontology_collection()

---

convert_network *Convert between supported network representations*

Description

This is the central object-to-object conversion interface. Use import_network() for files and tabular edge data, and read_network() for graph file formats handled by igraph, and export_network() to write third-party graph formats.

Usage

```
convert_network(
  from,
  to = c("igraph", "sparse", "legacy"),
  case_sensitive = FALSE
)
```

Arguments

from	An igraph object, sparse matrix, indexed sparse network, or legacy NEArender network.
to	Output representation: "igraph", "sparse", or "legacy".
case_sensitive	Preserve identifier case.

Value

A normalized network in the requested representation.

See Also

as_igraph_network(), as_sparse_network(), as_nearender_network(), import_network(), read_network(), export_network()

Examples

```
graph <- import_network(data.frame(from = "A", to = "B"),
  species = NULL, input_id_format = "custom", verbose = FALSE)
sparse <- convert_network(graph, "sparse")
legacy <- convert_network(sparse, "legacy")
legacy$links
```

`download_database` *Download a fixed-version curated database*

Description

Downloads are explicit, versioned, and cached. Existing files are reused unless `force = TRUE`. A catalog entry may comprise several jointly versioned and verified artifacts. No analytical function calls this function.

Usage

```
download_database(
  id,
  version = NULL,
  cache_dir = NULL,
  force = FALSE,
  quiet = FALSE,
  accept_license = FALSE
)
```

Arguments

`id`	Catalog identifier from `available_databases()`.
`version`	Database release specified by the catalog. By default, the release associated with `id` is used.
`cache_dir`	Cache root. Download functions require an explicit path; read-only loaders use the package user cache when it is `NULL`.
`force`	Replace existing cached artifacts.
`quiet`	Suppress download progress.
`accept_license`	Confirm that you reviewed the catalog entry's source terms. Required only when `requires_acceptance` is true.

Value

The normalized cached artifact path, or a named vector of paths for a grouped database, invisibly.

See Also

`available_databases()`, `load_database()`

Examples

```
cache <- file.path(tempdir(), "NEArender-cache")
## Not run:
download_database("reactome_pathways", cache_dir = cache)

## End(Not run)
```

`download_geneset_database` *Download a curated gene-set database*

Description

Type-specific wrapper around `download_database()`.

Usage

```
download_geneset_database(
  id,
  version = NULL,
  cache_dir = NULL,
  force = FALSE,
  quiet = FALSE,
  accept_license = FALSE
)
```

Arguments

`id`	Catalog identifier from `available_databases()`.
`version`	Database release specified by the catalog. By default, the release associated with `id` is used.
`cache_dir`	Cache root. Download functions require an explicit path; read-only loaders use the package user cache when it is `NULL`.
`force`	Replace existing cached artifacts.
`quiet`	Suppress download progress.
`accept_license`	Confirm that you reviewed the catalog entry's source terms. Required only when `requires_acceptance` is true.

Value

The downloaded artifact path or paths, invisibly.

See Also

`available_databases()`, `load_geneset_database()`

Examples

```
## Not run:
download_geneset_database("reactome_pathways")
reactome <- load_geneset_database("reactome_pathways")

## End(Not run)
```

`download_ndex_network` *Download an NDEx network snapshot*

Description

Download one arbitrary NDEx network in native CX2 (NDEx 3) or CX (NDEx 2) format. The current server summary is recorded first; the raw response is written to a temporary file, checked for the requested exchange-format marker and against the advertised byte count, checksummed, and atomically installed in a snapshot cache. Artifacts up to 10 MiB are also fully parsed as JSON before installation; larger artifacts are fully parsed when loaded. Repeated calls reuse a verified snapshot unless `force = TRUE`.

Usage

```
download_ndex_network(
  resource,
  format = c("cx2", "cx"),
  cache_dir = NULL,
  force = FALSE,
  quiet = FALSE,
  max_bytes = 1e+09,
  server = NULL,
  access_key = NULL,
  username = NULL,
  password = NULL,
  token = NULL,
  timeout = 300
)
```

Arguments

`resource`	One NDEx network UUID, NDEx network URL, or one row returned by `search_ndex_networks()`.
`format`	Native exchange format. CX2 is the current default; CX is provided for compatibility with clients that require the v2 representation.
`cache_dir`	Explicit cache root required for this writing operation.
`force`	Replace a valid cached snapshot.
`quiet`	Suppress download messages.

`max_bytes`	Maximum advertised and downloaded artifact size. The default is one gigabyte; use `Inf` only when the caller provides its own resource controls. The limit is enforced while response bytes are written.
`server`	NDEx server root or REST root. Defaults to the public NDEx server. Redirects are not followed. Internet-facing callers must allow-list trusted HTTPS origins rather than passing an arbitrary user-supplied URL.
`access_key`	Optional NDEx access key for an unlisted resource. It is used only for the request and is never written to a manifest.
`username`, `password`	Optional Basic authentication pair.
`token`	Optional bearer token. Do not combine with Basic credentials.
`timeout`	Positive request timeout in seconds.

Value

The normalized raw artifact path, invisibly.

See Also

`search_ndex_networks()`, `load_ndex_network()`, `manifest()`

Examples

```
## Not run:
path <- download_ndex_network(
  "f71ab602-97f0-11eb-9e72-0ac135e8bacf",
  cache_dir = file.path(tempdir(), "NEArender-cache")
)

## End(Not run)
```

`download_network_database`

Download a curated network database

Description

Type-specific wrapper around `download_database()`.

Usage

```
download_network_database(
  id,
  version = NULL,
  cache_dir = NULL,
  force = FALSE,
  quiet = FALSE,
  accept_license = FALSE
)
```

Arguments

id	Catalog identifier from available_databases().
version	Database release specified by the catalog. By default, the release associated with id is used.
cache_dir	Cache root. Download functions require an explicit path; read-only loaders use the package user cache when it is NULL.
force	Replace existing cached artifacts.
quiet	Suppress download progress.
accept_license	Confirm that you reviewed the catalog entry's source terms. Required only when requires_acceptance is true.

Value

The downloaded artifact path or paths, invisibly.

See Also

available_databases(), load_network_database()

Examples

```
## Not run:
download_network_database("string12_human")
string <- load_network_database("string12_human", n_top = 1000000)

## End(Not run)
```

---

download_ontology_collection

Download a GO or DO collection snapshot

Description

Explicitly download the release files required for one ontology collection, validate their formats, checksum them, and atomically install an immutable snapshot. Analytical functions never call this function implicitly.

Usage

```
download_ontology_collection(
  id = NULL,
  release = "current",
  cache_dir = NULL,
  force = FALSE,
  quiet = FALSE,
  channels = NULL,
```

```
  species = NULL,
  max_bytes = 2e+09,
  timeout = 1800
)
```

Arguments

id	Collection identifier from available_ontology_collections(). The shortcuts "DO" and "GO" select species, human by default. Use the named species argument for another supported organism.
release	"current" or an ontology release date (YYYY-MM-DD).
cache_dir	Explicit cache root required for this writing operation.
force	Query and download the provider again even when a verified pointer for this release exists. Existing immutable snapshots are kept.
quiet	Suppress progress messages.
channels	DO association channels. Integrated membership is always acquired. Ignored for GO.
species	Optional species code, taxon identifier, or scientific name.
max_bytes	Maximum total bytes downloaded in this call.
timeout	Per-file transfer timeout in seconds.

Details

GO snapshots pair go-basic.obo with one official organism GAF. DO snapshots pair doid.obo with the integrated DISEASES table and, by default, all three channel-specific tables. A dated DO release pins the ontology; the independently released DISEASES files are identified by their checksums and HTTP metadata in the same manifest.

Value

A named character vector of verified artifact paths, invisibly.

See Also

available_ontology_collections(), load_ontology_collection()

---

download_resource *Download an author-released network resource*

Description

Downloads occur only when this function, or prepare_resource(), is called explicitly. Files are cached below the supplied directory.

Usage

```
download_resource(spec, cache_dir, allow_large = FALSE, quiet = FALSE)
```

Arguments

spec	Resource identifier from network_catalog() or a specification returned by string_channel_spec() or humanbase_spec().
cache_dir	Explicit cache root.
allow_large	Confirm a catalog entry marked as a large download.
quiet	Suppress download progress.

Value

A named character vector of cached artifact paths, invisibly.

---

ear *Enrichment analysis on a rich, functionally biased gene/protein universe*

Description

EAR compares the enrichment score for each observed altered gene set (AGS) with scores from n_perm equally sized random sets sampled without replacement from the same eligible gene universe. The finite-sample corrected empirical probability is (1 + n_exceed) / (n_perm + 1), where n_exceed is the number of random scores greater than or equal to the observed score.

Usage

```
ear(
  universe,
  n_perm = 100L,
  ags,
  fgs,
  net = NULL,
  method = c("nea", "ora"),
  renderer_args = list(),
  seed = NULL,
  case_sensitive = FALSE,
  na_replace = 0,
  n_cores = 1L,
  verbose = TRUE,
  network_nodes_only = FALSE
)
```

Arguments

universe	Character vector of genes on which the observed sets are based and that are eligible for random controls.
n_perm	Positive integer number of controls per AGS.
ags	Named list of altered gene sets.
fgs	Named list of functional gene sets.
net	Network used for NEA. It is required when method = "nea".
method	Renderer used for calibration: "nea" or "ora".
renderer_args	Named list of additional arguments forwarded to the selected renderer. EAR-managed inputs and options cannot be overridden.
seed	Optional integer seed for reproducible control sampling.
case_sensitive	Use case-sensitive identifier matching.
na_replace	Value used for zero observed NEA link counts.
n_cores	Positive integer passed to the selected renderer. ORA distributes functional-set rows. NEA accepts the value for consistency, although EAR does not request member matrices and its sparse calculations remain single-process matrix operations.
verbose	Report progress with message().
network_nodes_only	For NEA, whether to restrict the random-control universe to genes present as network nodes. The default, FALSE, retains every gene in the submitted universe; genes absent from the network can occur in AGSs and controls but contribute no network edges. When TRUE, the universe is intersected with the network nodes and every AGS member must occur in that restricted universe.

Details

Calibration uses the upper tail: an empirical p-value counts random scores greater than or equal to the observed score. Q-values are adjusted separately within each AGS column. AGSs of different sizes use separate random reference banks. For NEA, the submitted gene universe is used by default even when some genes are absent from the network. Set network_nodes_only = TRUE to reproduce network-restricted sampling.

Value

A nearender_result whose master score is the upper-tail empirical probability matrix p.ear, available identically in scores$p.ear and statistics$p.ear. statistics also contains the ordinary renderer probabilities as p.trivial, their adjusted values as q.trivial, the empirical adjusted values as q.ear, and every other ordinary renderer statistic with a .trivial suffix. EAR settings are recorded in parameters; reference-bank information is recorded in diagnostics.

See Also

nea.render(), ora.render(), samples2ags()

Examples

```
data("NEArender.test")
ags <- NEArender.test$AGS$CCLE[1]
fgs <- NEArender.test$FGS$Hallmarks[1:2]
universe <- unique(c(unlist(ags), unlist(fgs), paste0("control", 1:150)))
calibrated <- ear(universe, n_perm = 9, ags = ags, fgs = fgs,
  method = "ora", seed = 1, verbose = FALSE)
calibrated$scores$p.ear
calibrated$statistics$p.trivial
```

---

export_network *Export a network through igraph*

Description

GraphML and GML are recommended when edge weights or other attributes must be retained. Simpler edge-list formats may discard attributes.

Usage

```
export_network(
  network,
  path,
  format = c("graphml", "gml", "ncol", "lgl", "pajek", "edgelist"),
  case_sensitive = TRUE,
  ...
)
```

Arguments

network	An igraph, legacy NEArender, or indexed sparse NEArender network graph.
path	Output graph file.
format	A format passed to the igraph writer.
case_sensitive	Preserve identifier case during conversion. When this argument is omitted for a normalized object, its recorded case policy is retained.
...	Additional arguments passed to igraph::write_graph().

Value

path, invisibly.

See Also

read_network(), import_network(), convert_network()

Examples

```
graph <- import_network(data.frame(from = "A", to = "B"),
  species = NULL, input_id_format = "custom", verbose = FALSE)
path <- tempfile(fileext = ".graphml")
export_network(graph, path)
file.exists(path)
unlink(path)
```

---

filter_ontology_collection

Filter an ontology collection without losing synchronized views

---

Description

Filter genes, terms, GO evidence codes, or DO confidence channels and rebuild all collection views. Derived counts, information content, score summaries, and evidence fractions are recalculated.

Usage

```
filter_ontology_collection(
  x,
  term_id = NULL,
  aspect = NULL,
  evidence_include = NULL,
  evidence_exclude = NULL,
  min_scores = NULL,
  score_mode = c("all", "any"),
  universe = NULL,
  min_members = NULL,
  max_members = NULL,
  min_depth = NULL,
  max_depth = NULL,
  min_structural_ic = NULL,
  min_annotation_ic = NULL,
  drop_empty = FALSE
)
```

Arguments

x	A nearender_ontology_collection.
term_id	Optional ontology term identifiers to retain.
aspect	Optional GO aspects (BP, MF, or CC) to retain.
evidence_include	Optional GO evidence-code allowlist.
evidence_exclude	Optional GO evidence-code denylist, for example "IEA". A membership remains when another allowed code supports it.

min_scores	Optional named numeric vector of minimum DO channel scores. Names may include or omit the _score suffix.
score_mode	Require "all" supplied channel thresholds or "any".
universe	Optional gene identifiers to retain.
min_members, max_members	Optional member-count bounds.
min_depth, max_depth	Optional bounds applied to minimum DAG depth.
min_structural_ic, min_annotation_ic	Optional information-content bounds.
drop_empty	Remove zero-member terms from the term table.

Value

A new nearender_ontology_collection.

---

glcm.net *Construct network grey-level co-occurrence matrices (GLCMs)*

Description

Construct network grey-level co-occurrence matrices (GLCMs)

Usage

```
glcm.net(
  mtr,
  net,
  rnm,
  t_level = 3,
  min_n = 5,
  by = c("non", "row", "col"),
  scope = c("within-pathway", "boundary-expanded")
)
```

Arguments

mtr	Numeric gene-by-sample matrix.
net	Network with a named adjacency list in links.
rnm	Genes defining one functional gene set or pathway.
t_level	Number of binary grey-level bits.
min_n	Minimum number of mapped genes required for the pathway. The function returns NULL below this threshold; texture.render() retains the pathway row and fills it with NA scores.
by	Rank-normalization direction.
scope	Graph scope: "within-pathway" uses within-set edges; "boundary-expanded" also uses boundary edges to one-hop neighbors.

Value

A named list with one normalized square GLCM per sample. Matrix cells contain symmetrized co-occurrence probabilities and sum to one. Returns NULL when fewer than min_n pathway genes map; an empty list indicates insufficient scoped edges.

See Also

haralick.net(), texture.render()

Examples

```
profiles <- matrix(c(1, 5, 2, 4, 3, 3, 4, 2), nrow = 4,
  dimnames = list(letters[1:4], c("sample_1", "sample_2")))
net <- list(links = list(a = "b", b = c("a", "c"),
  c = c("b", "d"), d = "c"), Ntotal = 3)
glcm <- glcm.net(profiles, net, c("a", "b", "c"),
  min_n = 3, scope = "within-pathway")
names(glcm)
```

---

gsva.render *Render continuous omics profiles to pathway scores with GSVA methods*

Description

Render continuous gene or protein sample profiles as pathway-level score matrices using methods provided by the GSVA package.

Usage

```
gsva.render(
  fgs,
  mtr,
  method = c("gsva", "ssgsea", "zscore", "plage"),
  min_ngenes = 1L,
  max_ngenes = Inf,
  method_parameters = list(),
  case_sensitive = FALSE,
  verbose = TRUE
)
```

Arguments

fgs	Named list of functional gene sets.
mtr	Numeric gene-by-sample expression matrix.
method	Unique methods selected from "gsva", "ssgsea", "zscore", and "plage".
min_ngenes	Minimum gene set size passed to GSVA.

max_ngenes	Maximum gene set size passed to GSVA.
method_parameters	Named list of method-specific argument lists, forwarded to the corresponding *Param() constructor.
case_sensitive	Use case-sensitive gene identifier matching.
verbose	Report progress from GSVA.

Details

Every selected GSVA-family method is run separately on the same gene-by-sample matrix and FGS list. Method-specific options are supplied in method_parameters. These are pathway activity scores; this wrapper does not calculate p- or q-values.

Value

A nearender_result. scores contains one pathway-by-sample matrix per selected method, in request order. statistics contains the same master score matrices under the same names. diagnostics$mapped_fgs_sizes records the number of input-matrix genes mapped to each FGS. inputs records pathway and sample names, and parameters records size limits and method-specific arguments.

See Also

texture.render(), spectral.render(), ora.render()

Examples

```
profiles <- matrix(seq_len(24), nrow = 6,
  dimnames = list(letters[1:6], paste0("sample_", 1:4)))
fgs <- list(first = c("a", "b", "c"), second = c("d", "e", "f"))
if (interactive() && requireNamespace("GSVA", quietly = TRUE)) {
  result <- gsva.render(fgs, profiles, method = "zscore",
    verbose = FALSE)
  result$scores$zscore
}
```

---

haralick.matrix *Compatibility wrapper for texture rendering*

Description

haralick.matrix() forwards to texture.render() and emits a deprecation warning. Its former set-selection and combined all arguments are not part of this interface.

Usage

```
haralick.matrix(
  fgs,
  mtr,
  net,
  t_level = 3,
  min_ngenes = 5,
  norm_dir = c("row", "col", "non"),
 features = c("asm", "con", "cor", "den", "dva", "ent", "f12", "f13", "idm", "pro",
    "sav", "sen", "sha", "sva", "var"),
  scope = c("within-pathway", "boundary-expanded"),
  case_sensitive = FALSE,
  n_cores = 1L,
  verbose = TRUE
)
```

Arguments

fgs	Named list of functional gene sets whose subnetworks are extracted from the global network. Subset it before calling when needed.
mtr	Numeric gene-by-sample matrix.
net	Network with a named adjacency list in `links`.
t_level	Number of binary grey-level bits.
min_ngenes	Minimum mapped genes per set.
norm_dir	Rank-normalization direction.
features	Unique requested Haralick feature identifiers. The default explicitly lists all 15 implemented features.
scope	Graph scope: `"within-pathway"` uses within-set edges; `"boundary-expanded"` also uses boundary edges to one-hop neighbors.
case_sensitive	Use case-sensitive identifier matching.
n_cores	Positive integer number of worker processes. Independent functional gene sets are distributed across workers when greater than one.
verbose	Report progress with `message()`.

Value

A `nearender_result` returned by `texture.render()`.

See Also

`texture.render()`, `haralick.net()`, `glcm.net()`

`haralick.net` *Calculate Haralick texture features for one functional gene set*

Description

Calculate texture features from sample-specific grey-level co-occurrence matrices (GLCMs) for a single functional gene set. `glcm.net()` constructs this input; `texture.render()` calls both low-level functions across multiple functional gene sets and assembles the pathway-by-sample result matrices.

Usage

```
haralick.net(
  x,
  bind_range = FALSE,
 features = c("asm", "con", "cor", "den", "dva", "ent", "f12", "f13", "idm", "pro",
    "sav", "sen", "sha", "sva", "var")
)
```

Arguments

`x`	Named list of square GLCMs, one per sample, for one functional gene set.
`bind_range`	Append each feature's range across samples.
`features`	Unique feature identifiers. The default explicitly lists all 15 implemented features.

Details

The calculation follows the established `wvtool` indexing convention. Although discrete grey levels are labelled `0` through `G - 1` in the formulas, the `spq` index for sum-based features is formed from R matrix indices `1` through `G`. Consequently, `sav` is two units above a strictly zero-based implementation, and `sva` uses the corresponding scale. This convention is retained for numerical consistency.

Value

A numeric feature-by-sample matrix in requested feature order. When `bind_range = TRUE`, a final `rng` column gives each feature's across-sample range. Returns `NULL` for an empty GLCM list.

Haralick features

`features` accepts the following identifiers:

`asm` angular second moment (energy)

`con` contrast

`cor` correlation

`den` difference entropy

`dva` difference variance

`ent` entropy

`f12` first information measure of correlation

`f13` second information measure of correlation

`idm` inverse difference moment (homogeneity)

`pro` cluster prominence

`sav` sum average

`sen` sum entropy

`sha` cluster shade

`sva` sum variance

`var` variance

References

Haralick RM, Shanmugam K, Dinstein I (1973). doi:10.1109/TSMC.1973.4309314. Kobayashi K et al. (2015). doi:10.1007/s1008601515076. Alexeyenko A (2026). doi:10.64898/2026.07.12.737672.

See Also

`glcm.net()`, `texture.render()`, `haralick.matrix()`

Examples

```
p1 <- matrix(c(0.4, 0.1, 0.1, 0.4), 2)
p2 <- matrix(c(0.25, 0.25, 0.25, 0.25), 2)
haralick.net(list(sample_1 = p1, sample_2 = p2),
  features = c("asm", "con", "idm"))
```

---

humanbase_spec *Describe a tissue-specific HumanBase network*

Description

Describe a tissue-specific HumanBase network

Usage

```
humanbase_spec(tissue, version = c("giant", "mage"))
```

Arguments

`tissue`	HumanBase tissue name.
`version`	GIANT or MAGE network generation.

Value

A resource specification accepted by `prepare_resource()`.

import.net *Legacy network importer*

Description

`import.net()` is retained for compatibility with legacy NEArender scripts. New code should use `import_network()`, which has explicit input-column and output-format arguments.

Usage

```
import.net(
  tbl,
  lowercase = 1,
  col_1 = 1,
  col_2 = 2,
  col_score = 3,
  min_score = NA,
  n_top = NA,
  echo = 1
)
```

Arguments

`tbl`, `lowercase`, `col_1`, `col_2`, `col_score`, `min_score`, `n_top`, `echo`
: Legacy arguments.

Value

A legacy NEArender network list with `links` and `Ntotal`.

See Also

`import_network()`

import_do_collection *Import a Disease Ontology collection from local release files*

Description

Build an unfiltered, feature-rich Disease Ontology collection from an OBO ontology and local DISEASES association tables. The integrated channel defines membership. Optional channel confidences are joined to those memberships and duplicate channel rows are represented by their maximum confidence. DISEASES already backtracks associations to ontology parents, so this function never propagates membership itself.

Usage

```
import_do_collection(
  ontology,
  integrated,
  experiments = NULL,
  knowledge = NULL,
  textmining = NULL,
  case_sensitive = FALSE,
  include_external_terms = FALSE,
  species = NULL,
  verbose = FALSE
)
```

Arguments

ontology	Local Disease Ontology OBO file.
integrated	Local human_disease_integrated_full.tsv file.
experiments	Optional experiments-channel TSV file.
knowledge	Optional knowledge-channel TSV file.
textmining	Optional text-mining TSV file.
case_sensitive	Preserve the submitted gene-name case.
include_external_terms	Retain non-DO terms such as ICD hierarchy nodes. Their ontology graph features are missing.
species	Optional species identifier. When supplied, it must resolve to human ("hsa"); the DISEASES source used here is human-specific.
verbose	Report import progress and dimensions.

Value

A nearender_ontology_collection.

---

import_gene_sets *Import altered or functional gene sets*

Description

Read a tabular altered gene set (AGS) or functional gene set (FGS) source into the named-list representation used by NEArender. In grouped layout, each row associates one gene or protein identifier with a sample, pathway, or other set identifier. Single-set layout combines all genes into one set; single-gene layout creates one set per distinct gene.

Usage

```
import_gene_sets(
  source,
  gene_column = 1L,
  set_column = 2L,
  layout = c("grouped", "single_set", "single_gene"),
  set_name = "user_set",
  header = FALSE,
  sep = "\t",
  comment_char = "",
  case_sensitive = FALSE,
  mask = NULL,
  species,
  input_id_format = master_id_format(species),
  id_mappings = NULL,
  ambiguous = c("error", "drop", "first", "expand"),
  unmapped = c("keep", "drop", "error")
)

import.gs(
  tbl,
  lowercase = 1,
  col_gene = 2,
  col_set = 3,
  gs_type = "",
  header = FALSE
)
```

Arguments

source	File path, URL, connection, matrix, or data frame.
gene_column	Name or one-based index of the gene/protein column.
set_column	Name or one-based index of the set column. Required for `layout = "grouped"` and ignored by the other layouts.
layout	Input organization: `"grouped"`, `"single_set"`, or `"single_gene"`.
set_name	Name used for `layout = "single_set"`.
header	Whether a text source has a header row. Ignored for a matrix or data frame.
sep	Field separator used for text sources. Ignored for a matrix or data frame.
comment_char	Comment marker used for text sources. The default disables comment processing. Ignored for a matrix or data frame.
case_sensitive	Preserve case when matching gene identifiers.
mask	Optional POSIX regular expression passed to `grepl()` to retain set names in grouped layout. For example, `"^keep_"` matches a prefix, `"_pathway$"` matches a suffix, and `"first|second"` matches either alternative. Escape regular-expression metacharacters when matching them literally. The default, `NULL`, retains every set.

`species` Required species identifier. Accepts any scientific/common name, package species ID, three-letter KEGG style organism ID, or taxonomy ID listed by `available_id_formats()`. Use explicit `NULL` only for unsupported species or non-biological data.

`input_id_format`
Submitted identifier format. Defaults to `master_id_format()` for `species`; non-master input must name a format listed by `available_id_formats()`. Required when `species = NULL`.

`id_mappings` Optional object from `load_id_mappings()`. When supplied, identifiers are converted to its species master format.

`ambiguous`, `unmapped`
Mapping policies passed to `map_identifiers()`.

`tbl`, `lowercase`, `col_gene`, `col_set`, `gs_type`
Legacy arguments. The modern `header` argument is shared by both interfaces.

Details

Gene identifiers are unique within each set and are matched case-insensitively by default. Set names retain their submitted spelling. In `layout = "grouped"`, each distinct `set_column` value creates one list element. `layout = "single_set"` combines the selected gene column into one set named by `set_name`. `layout = "single_gene"` creates one single-member set per distinct identifier; this can be used to treat network genes as FGSs in NEA, whereas overlap-based ORA has no analogous edge-enrichment use.

Value

A named list whose elements contain unique gene identifiers, with normalization metadata available through `normalization_info()`.

See Also

`save_gs_list()`, `import_kegg_pathways()`, `load_geneset_database()`, `samples2ags()`, `mutations2ags()`

Examples

```
table <- data.frame(gene = c("A", "B", "C"),
  pathway = c("first", "first", "second"))
import_gene_sets(table, "gene", "pathway", species = "human")
import_gene_sets(table, "gene", "pathway", mask = "^first$",
  species = "human")
one_column <- data.frame(gene = c("A", "B", "C"))
import_gene_sets(one_column, layout = "single_set", set_name = "panel",
  species = "human")
```

`import_go_collection` *Import a Gene Ontology collection from local OBO and GAF files*

Description

Build a compact GO collection. Positive annotations are propagated through `is_a` and `part_of` by default after excluding IEA. Pass `evidence_exclude = NULL` to retain every positive evidence code. `NOT` and `ND` records are retained as audit information but never define membership.

Usage

```
import_go_collection(
  ontology,
  annotations,
  species = NULL,
  propagate = TRUE,
  relations = c("is_a", "part_of"),
  evidence_exclude = "IEA",
  case_sensitive = FALSE,
  verbose = FALSE
)
```

Arguments

`ontology`	Local `go-basic.obo` file.
`annotations`	Local GAF file, optionally gzip-compressed.
`species`	Required NEArender species identifier describing the symbol namespace. It does not trigger identifier mapping or downloads.
`propagate`	Propagate positive direct annotations to ontology parents.
`relations`	Ontology relations used for graph metrics and propagation. `is_a` and `part_of` are the safe defaults. Regulation relations may be selected explicitly; inverse `has_part` propagation is rejected.
`evidence_exclude`	Positive GO evidence codes excluded before ancestor propagation. The default is `"IEA"`; use `NULL` to retain all codes.
`case_sensitive`	Preserve submitted gene-symbol case.
`verbose`	Report import progress and dimensions.

Value

A compact `nearender_ontology_collection` containing `fgs`, named evidence-code vectors, membership evidence, and all active GO terms.

`import_kegg_pathways` *Import a licensed KEGG pathway table*

Description

Import an organism-specific, user-supplied KEGG pathway membership table. The expected layout has Ensembl gene identifiers in column 1, gene symbols in column 2, and pathway names in column 3, as in KEGG FTP-derived exports. NEArender never downloads or redistributes this licensed input.

Usage

```
import_kegg_pathways(
  source,
  species,
  gene_column = c("symbol", "ensembl"),
  id_mappings = NULL,
  kegg_release = NA_character_,
  case_sensitive = FALSE,
  ambiguous = c("error", "drop", "first", "expand"),
  unmapped = c("keep", "drop", "error")
)
```

Arguments

`source`	Path to a user-supplied KEGG pathway membership table.
`species`	Required species identifier accepted by `available_id_formats()`.
`gene_column`	Identifier column to use: `"symbol"` selects column 2 and `"ensembl"` selects column 1.
`id_mappings`	Optional object from `load_id_mappings()`. Required when `gene_column = "ensembl"`.
`kegg_release`	Optional KEGG release or acquisition-date label recorded in provenance.
`case_sensitive`	Preserve identifier case.
`ambiguous, unmapped`	Identifier-mapping policies used for Ensembl IDs.

Details

Gene symbols are the default because they may remain usable when KEGG's Ensembl cross-references lag the current identifier release. Select `gene_column = "ensembl"` to use column 1 explicitly; a corresponding `id_mappings` object is then required to convert the identifiers to the species master namespace.

Value

A named pathway list with normalization and source provenance available through `normalization_info()`.

See Also

import_gene_sets(), available_databases(), load_id_mappings()

Examples

```
path <- tempfile(fileext = ".rno")
writeLines(c(
  "ENSRNOG00000000001\tTp53\tp53 signaling pathway",
  "ENSRNOG00000000002\tMdm2\tp53 signaling pathway"), path)
pathways <- import_kegg_pathways(path, species = "rat",
  gene_column = "symbol", kegg_release = "example")
unlink(path)
pathways
```

---

import_network *Import an undirected interaction network*

Description

Read a global undirected network from a tabular edge list, import a square adjacency matrix, or normalize a supported network object. Reciprocal and duplicate edges are collapsed and self-links are removed. When scored duplicate rows disagree, a warning reports their count and examples. The highest edge score is retained, or the highest absolute score when an absolute-value selector is used. The selected source edge-score column becomes the single canonical igraph `weight` attribute or adjacency-aligned legacy weights. In a user-supplied scored table, a source score of zero is interpreted as no edge and skipped. Existing network objects and curated adapters preserve zero edge weights.

Usage

```
import_network(
  source,
  from_column = 1L,
  to_column = 2L,
  score_column = NULL,
  min_score = NULL,
  n_top = NULL,
  greedy = TRUE,
  header = FALSE,
  sep = "\t",
  comment_char = "",
  case_sensitive = FALSE,
  verbose = TRUE,
  output_network_format = c("igraph", "sparse", "legacy"),
  species,
  input_id_format = master_id_format(species),
  id_mappings = NULL,
```

```
  ambiguous = c("error", "drop", "first", "expand"),
  unmapped = c("keep", "drop", "error"),
  is_matrix = FALSE,
  min_abs_score = NULL,
  n_abs_top = NULL
)
```

Arguments

source	File path, URL, connection, matrix, data frame, or supported network object.
from_column, to_column	Names or one-based indices of edge endpoint columns.
score_column	Optional name or one-based index of the source edge score column. Its numeric values are stored as edge weights. If NULL, imported edges are unweighted.
min_score	Optional minimum retained edge score.
n_top	Optional number of highest-scoring unique edges to retain.
greedy	Boundary-tie policy used with n_top or n_abs_top. The default, TRUE, includes every edge tied at the selection boundary; FALSE keeps only edges strictly above it; and NULL returns exactly the requested number of edges by filling the tie in alphanumeric canonical-endpoint order. Use greedy = NULL when equal network sizes are required for consistent benchmarks.
header	Whether a text source has a header row. Ignored for a matrix or data frame.
sep	Field separator used for text sources. Ignored for a matrix or data frame.
comment_char	Comment marker used for text sources. The default disables comment processing. Ignored for a matrix or data frame.
case_sensitive	Preserve case when matching node identifiers.
verbose	Report the imported node and edge counts.
output_network_format	Return an igraph, sparse, or legacy NEArender network.
species	Required species identifier. Accepts any scientific/common name, package species ID, three-letter KEGG style organism ID, or taxonomy ID listed by available_id_formats(). Use explicit NULL only for unsupported species or non-biological data.
input_id_format	Submitted identifier format. Defaults to master_id_format() for species; non-master input must name a format listed by available_id_formats(). Required when species = NULL.
id_mappings	Optional object from load_id_mappings(). When supplied, both endpoints are converted to its species master format.
ambiguous, unmapped	Mapping policies passed to map_identifiers().
is_matrix	Treat source as a square undirected adjacency matrix rather than as a tabular edge list. Row and column names must be identical and non-empty, and the matrix must be symmetric. Its values become edge weights; the diagonal is ignored and zero off-diagonal entries are treated as absent edges. from_column, to_column, and score_column may be omitted or explicitly set to NULL in this mode.

`min_abs_score` Optional minimum retained absolute edge score. This is useful for signed adjacency matrices such as correlation matrices. The signed value is preserved as the edge weight.

`n_abs_top` Optional number of unique edges with the highest absolute scores to retain. Signed edge weights are preserved. The `greedy` boundary-tie policy also applies to this option. Only one of `min_score`, `min_abs_score`, `n_top`, and `n_abs_top` may be supplied.

Details

An edge weight is stored metadata; it is not automatically a transition probability. Renderers such as `rwr.render()` and `pagerank.render()` use topology alone unless `use_edge_weights = TRUE`. A transition probability is derived, when requested, by normalizing the non-negative outgoing edge weights.

NEArender does not create directed networks. With `is_matrix = TRUE`, `source` must therefore be a symmetric adjacency matrix. Asymmetric matrices are rejected rather than interpreted as directed networks. For dense matrices, edge-score filtering is applied while the matrix is scanned, so excluded cells are never materialized as edge rows. Top-N selection uses bounded chunks to determine its score boundary first. After a matrix import, temporary edge tables are collected. On Linux systems using glibc, free allocator pages are also returned to the operating system so repeated imports do not retain their peak RSS.

Pathway Commons aggregates interaction evidence from multiple curated resources and records supporting publications for most edges. The catalogued `pathwaycommons_hgnc` adapter derives a PubMed-count edge score: the number of distinct supporting publications plus a 0.5 pseudocount. Relations without a linked PubMed record therefore remain edges with weight 0.5. With `load_network_database()`, `min_score = 1.5` requires at least one publication and `min_score = 2.5` requires at least two.

Value

A checked and normalized undirected igraph object by default, a sparse network, or a named-adjacency-list network. Weighted igraph output has one numeric `weight` edge attribute; sparse output stores binary `adjacency` separately from `weights`; legacy output stores named adjacency-aligned `weights`. Identifier and repair metadata are available through `normalization_info()`.

See Also

`available_databases()`, `load_network_database()`, `read_network()`, `export_network()`, `convert_network()`

Examples

```
edges <- data.frame(from = c("A", "B", "B"), to = c("B", "A", "C"),
  score = c(0.8, 0.9, 0.7))
import_network(edges, "from", "to", "score", verbose = FALSE,
  species = "human")
correlation <- matrix(c(0, 0.8, -0.4, 0.8, 0, 0.2, -0.4, 0.2, 0),
  nrow = 3, dimnames = list(c("A", "B", "C"), c("A", "B", "C")))
import_network(correlation, is_matrix = TRUE, min_abs_score = 0.4,
  verbose = FALSE, species = NULL, input_id_format = "custom")
```

```
import_network(correlation, is_matrix = TRUE, n_abs_top = 2,
  greedy = NULL, verbose = FALSE, species = NULL,
  input_id_format = "custom")
```

`load_database` *Load a cached curated database*

Description

Parsing is offline and verifies every cached artifact before use. Large network adapters parse in chunks and may require an explicit edge filter. For `pathwaycommons_hgnc`, the edge weight is the number of distinct supporting PubMed publications plus a 0.5 pseudocount. Thus uncited curated relations remain edges with weight 0.5; use `min_score = 2.5` to require at least two publications. Selected source edge scores are stored as the canonical network edge weight.

Usage

```
load_database(
  id,
  version = NULL,
  cache_dir = NULL,
  case_sensitive = FALSE,
  min_score = NULL,
  n_top = NULL,
  greedy = TRUE,
  map_to_symbols = NULL,
  chunk_size = 100000L,
  id_mappings = NULL,
  ambiguous = c("drop", "error", "first", "expand"),
  unmapped = c("drop", "keep", "error"),
  output_network_format = c("igraph", "sparse", "legacy"),
  network_format = NULL,
  verbose = TRUE
)
```

Arguments

`id`	Catalog identifier from `available_databases()`.
`version`	Database release specified by the catalog. By default, the release associated with `id` is used.
`cache_dir`	Cache root. Download functions require an explicit path; read-only loaders use the package user cache when it is `NULL`.
`case_sensitive`	Preserve identifier case.
`min_score`	Optional minimum network edge score.
`n_top`	Optional number of highest-scoring unique network edges.

greedy — Boundary-tie policy used with n_top. The default, TRUE, includes every edge tied at the score of the n_topth edge and can therefore return more than n_top edges. FALSE keeps only edges strictly above that boundary score and can return fewer edges. NULL returns exactly n_top edges by filling a boundary tie in alphanumeric canonical-endpoint order. The strict-size NULL mode is useful when benchmarked networks must contain consistent edge counts.

map_to_symbols — Convert network identifiers to the species master identifier format. The default, NULL, enables conversion for networks and disables this network-only option for gene-set resources. GeneMANIA uses its bundled release-matched mapping. STRING and FunCoup use a cached mapping prepared by prepare_id_mappings(). Set this to FALSE only to retain provider-native identifiers.

chunk_size — Number of network rows parsed per chunk.

id_mappings — Optional object from load_id_mappings() used to convert native STRING or FunCoup identifiers. When omitted, the loader uses the verified local mapping cache and tells the user to run prepare_id_mappings() first if it is absent.

ambiguous, unmapped — Identifier-mapping policies passed to import_network().

output_network_format — Return curated networks as igraph objects, sparse networks, or legacy NEArender adjacency lists. Gene set databases are unaffected.

network_format — Deprecated alias for output_network_format.

verbose — Report validation, parsing progress, and final object sizes.

Value

The loaded network or gene-set collection. Use manifest() to inspect its provenance and loading statistics.

See Also

available_databases(), download_database(), import_network(), import_gene_sets()

Examples

```
## Not run:
download_database("string12_human")
loaded <- load_database("string12_human", n_top = 1000000)
loaded
manifest(loaded)

## End(Not run)
```

load_geneset_database *Load a cached curated gene-set database*

Description

Type-safe wrapper around load_database() with a gene-set-specific signature.

Usage

```
load_geneset_database(
  id,
  version = NULL,
  cache_dir = NULL,
  case_sensitive = FALSE,
  verbose = TRUE
)
```

Arguments

id	Catalog identifier from available_databases().
version	Database release specified by the catalog. By default, the release associated with id is used.
cache_dir	Cache root. Download functions require an explicit path; read-only loaders use the package user cache when it is NULL.
case_sensitive	Preserve identifier case.
verbose	Report validation, parsing progress, and final object sizes.

Value

The loaded gene-set collection. Use manifest() for provenance.

See Also

download_geneset_database(), import_gene_sets()

Examples

```
## Not run:
pathways <- load_geneset_database("reactome_pathways")
length(pathways)

## End(Not run)
```

load_id_mappings *Load cached identifier mappings*

Description

Read a previously prepared identifier mapping without making a network request.

Usage

```
load_id_mappings(species, release = NULL, cache_dir = NULL)
```

Arguments

species	One supported scientific name, common name, package species ID, three-letter KEGG style organism ID, or NCBI taxonomy ID.
release	Concrete cached release. NULL or 'current' uses the most recently prepared release for the species.
cache_dir	Cache root. By default, this is `tools::R_user_dir("NEArender", which = "cache")`.

Value

A list containing long-form mappings, annotations, and the provenance manifest.

Examples

```
## Not run:
mappings <- load_id_mappings("human")
mappings$manifest$release

## End(Not run)
```

load_ndex_network *Load a cached or local NDEx network*

Description

Parse a previously downloaded CX2/CX artifact and project it into NEArender's simple undirected network model. Loading is strictly offline. Node identifiers come from the explicitly selected CX node attribute; direction, visual styles, and unselected attributes remain in the raw artifact rather than being guessed into the analytical network. Reciprocal and duplicate edges are collapsed by `import_network()`.

Usage

```
load_ndex_network(
  resource,
  species,
  input_id_format = master_id_format(species),
  cache_dir = NULL,
  snapshot = NULL,
  format = c("auto", "cx2", "cx"),
  node_key = "name",
  score_column = NULL,
  max_bytes = 1e+09,
  output_network_format = c("igraph", "sparse", "legacy"),
  min_score = NULL,
  n_top = NULL,
  greedy = TRUE,
  case_sensitive = FALSE,
  verbose = TRUE,
  id_mappings = NULL,
  ambiguous = c("error", "drop", "first", "expand"),
  unmapped = c("keep", "drop", "error"),
  server = NULL
)
```

Arguments

`resource`	One cached NDEx UUID, one catalog row, or a local CX/CX2 artifact path. A path need not reside in the NEArender cache.
`species`	Required species declaration passed to `import_network()`. Use explicit `NULL` for unsupported species or non-biological nodes.
`input_id_format`	Submitted node identifier format. It defaults to the species master format and must be explicit when `species = NULL`.
`cache_dir`	Cache root used when `resource` is a UUID or catalog row.
`snapshot`	Optional cached snapshot identifier. By default the most recently downloaded snapshot pointer is used.
`format`	Exchange format. `"auto"` detects a local file and prefers a cached CX2 snapshot over CX.
`node_key`	CX node identifier field. Use `"name"`, `"represents"`, `"id"`, or any scalar node attribute present in the network.
`score_column`	Optional scalar numeric CX edge attribute to retain as the canonical NEArender edge weight.
`max_bytes`	Maximum raw artifact size accepted for in-memory JSON parsing.
`output_network_format`	Return an igraph, sparse, or legacy NEArender network.
`min_score, n_top, greedy`	Edge filtering passed to `import_network()`.

case_sensitive	Preserve identifier case.
verbose	Report import dimensions.
id_mappings	Optional object from load_id_mappings().
ambiguous, unmapped	Identifier mapping policies passed to import_network().
server	NDEx server root or REST root. Defaults to the public NDEx server. Redirects are not followed. Internet-facing callers must allow-list trusted HTTPS origins rather than passing an arbitrary user-supplied URL.

Value

A normalized NEArender network with NDEx acquisition and projection provenance available through manifest().

See Also

download_ndex_network(), search_ndex_networks(), import_network()

Examples

```
## Not run:
download_ndex_network("f71ab602-97f0-11eb-9e72-0ac135e8bacf")
network <- load_ndex_network(
  "f71ab602-97f0-11eb-9e72-0ac135e8bacf",
  species = "human", node_key = "name"
)
manifest(network)

## End(Not run)
```

---

load_network_database *Load a cached curated network database*

Description

Type-safe wrapper around load_database() with a network-specific signature and master-identifier conversion enabled by default.

Usage

```
load_network_database(
  id,
  version = NULL,
  cache_dir = NULL,
  case_sensitive = FALSE,
  min_score = NULL,
  n_top = NULL,
```

```
  greedy = TRUE,
  map_to_symbols = TRUE,
  chunk_size = 100000L,
  id_mappings = NULL,
  ambiguous = c("drop", "error", "first", "expand"),
  unmapped = c("drop", "keep", "error"),
  output_network_format = c("igraph", "sparse", "legacy"),
  network_format = NULL,
  verbose = TRUE
)
```

Arguments

id	Catalog identifier from available_databases().
version	Database release specified by the catalog. By default, the release associated with id is used.
cache_dir	Cache root. Download functions require an explicit path; read-only loaders use the package user cache when it is NULL.
case_sensitive	Preserve identifier case.
min_score	Optional minimum network edge score.
n_top	Optional number of highest-scoring unique network edges.
greedy	Boundary-tie policy used with n_top. The default, TRUE, includes every edge tied at the score of the n_topth edge and can therefore return more than n_top edges. FALSE keeps only edges strictly above that boundary score and can return fewer edges. NULL returns exactly n_top edges by filling a boundary tie in alphanumeric canonical-endpoint order. The strict-size NULL mode is useful when benchmarked networks must contain consistent edge counts.
map_to_symbols	Convert network identifiers to the species master identifier format. The default, NULL, enables conversion for networks and disables this network-only option for gene-set resources. GeneMANIA uses its bundled release-matched mapping. STRING and FunCoup use a cached mapping prepared by prepare_id_mappings(). Set this to FALSE only to retain provider-native identifiers.
chunk_size	Number of network rows parsed per chunk.
id_mappings	Optional object from load_id_mappings() used to convert native STRING or FunCoup identifiers. When omitted, the loader uses the verified local mapping cache and tells the user to run prepare_id_mappings() first if it is absent.
ambiguous, unmapped	Identifier-mapping policies passed to import_network().
output_network_format	Return curated networks as igraph objects, sparse networks, or legacy NEArender adjacency lists. Gene set databases are unaffected.
network_format	Deprecated alias for output_network_format.
verbose	Report validation, parsing progress, and final object sizes.

Value

The loaded network. Use `manifest()` for provenance.

See Also

`download_network_database()`, `import_network()`

Examples

```
## Not run:
network <- load_network_database("string12_human", n_top = 1000000)
manifest(network)$filters

## End(Not run)
```

`load_ontology_collection`

Load a cached GO or DO collection

Description

Load and parse one previously downloaded ontology snapshot. This operation is offline; it never updates or downloads source data. The returned `nearender_ontology_collection` contains the synchronized `fgs`, `associations`, `membership`, and `terms` views and carries the verified snapshot manifest.

Usage

```
load_ontology_collection(
  id = NULL,
  snapshot = NULL,
  release = "current",
  cache_dir = NULL,
  channels = "integrated",
  species = NULL,
  ...
)
```

Arguments

`id`	Collection identifier from `available_ontology_collections()`. The shortcuts "DO" and "GO" select species, human by default. Use the named species argument for another supported organism.
`snapshot`	Optional immutable snapshot identifier. The pointer for `release` is used when omitted.
`release`	Release pointer selected when `snapshot` is omitted.
`cache_dir`	Cache root.

channels	DO channels to parse. The default, "integrated", yields the compact integrated-only collection. Use "all" or an explicit vector for optional channels already present in the cached snapshot. Ignored for GO.
species	Optional species code, taxon identifier, or scientific name.
...	Parsing and filtering arguments passed to import_go_collection() or import_do_collection().

Value

A nearender_ontology_collection.

See Also

download_ontology_collection(), import_go_collection(), import_do_collection()

---

manifest *Inspect a curated-database provenance manifest*

Description

Return the manifest attached to an object loaded by load_database(), load_network_database(), load_geneset_database(), load_ontology_collection(), or one of the local ontology importers. Objects without recorded provenance return NULL.

Usage

```
manifest(x)
```

Arguments

x	A network, gene-set, or ontology collection.

Value

A provenance manifest list, or NULL.

Examples

```
manifest(list(pathway = c("TP53", "MDM2")))
```

`map_identifiers` *Map identifiers to a species master namespace*

Description

Apply a mapping loaded by `load_id_mappings()` without network access. Mapping is deliberately explicit: callers select the input namespace and policies for ambiguous and unmapped identifiers before importing or preparing analytical objects. This public helper is also useful for mapping an identifier vector independently of an import operation.

Usage

```
map_identifiers(
  ids,
  mappings,
  from,
  ambiguous = c("error", "drop", "first", "expand"),
  unmapped = c("keep", "drop", "error"),
  case_sensitive = TRUE
)
```

Arguments

`ids`	Character vector of source identifiers.
`mappings`	A `nearender_id_mappings` object returned by `load_id_mappings()`.
`from`	Source identifier format listed by `available_id_formats()`.
`ambiguous`	Policy for source IDs with several master IDs: error, drop, keep the first sorted match, or expand to all matches.
`unmapped`	Policy for source IDs without a mapping: keep, drop, or error.
`case_sensitive`	Match source identifiers with case sensitivity.

Value

A character vector in the master namespace. With `ambiguous = "expand"`, one input may yield several output values. A `mapping_info` attribute records mapping counts and policies.

See Also

`prepare_id_mappings()`, `load_id_mappings()`

Examples

```
example_mappings <- structure(list(
  mappings = data.frame(id_format = "ensembl_gene_id",
    source_id = c("ENSG1", "ENSG2"), master_id = c("TP53", "MDM2")),
  manifest = list(id_formats = "ensembl_gene_id")),
```

```
  class = "nearender_id_mappings")
map_identifiers(c("ENSG1", "ENSG2"), example_mappings,
  from = "ensembl_gene_id")
```

`master_id_format` *Return the master identifier format for a species*

Description

Resolve any supported species alias and return the concise, user-facing name of its symbol-oriented master identifier format.

Usage

```
master_id_format(species)
```

Arguments

`species`	Scientific or common name, package species ID, three-letter KEGG style organism ID, or NCBI taxonomy ID accepted by `available_id_formats()`.

Value

One character value naming the species master format, such as `"HGNC"`, `"MGI"`, or `"RGD"`.

Examples

```
master_id_format("9606")
master_id_format("mouse")
```

`merge_networks` *Merge network topologies*

Description

Combine the edge unions of two or more networks after checking identifier compatibility. Edge scores are omitted by default. They can be retained only by explicit request when every input is weighted and their recorded score formats are compatible. Duplicate scored edges retain the maximum score.

Usage

```
merge_networks(
  nets,
  preserve_scores = FALSE,
  score_format = NULL,
  condition = "union",
  output_network_format = c("igraph", "sparse", "legacy"),
  case_sensitive = NULL
)
```

Arguments

nets	Named list containing at least two supported network objects.
preserve_scores	Preserve compatible input edge scores. The default, FALSE, creates an unweighted network.
score_format	Optional explicit score-format declaration. This is required with preserve_scores = TRUE when any scored input lacks score provenance. It does not override conflicting recorded formats.
condition	Edge-presence rule: "union", "intersection", or an integer giving the minimum number of distinct input networks in which an edge must occur.
output_network_format	Output representation.
case_sensitive	Identifier case policy. By default, reuse the common recorded policy of the inputs, or use case-insensitive matching when none is recorded.

Value

A merged network whose manifest() records all sources, score handling, and output counts.

See Also

reduce_network(), network_overlap(), manifest()

Examples

```
first <- import_network(data.frame(from = c("A", "B"), to = c("B", "C")),
  species = NULL, input_id_format = "custom", verbose = FALSE)
second <- import_network(data.frame(from = "C", to = "D"),
  species = NULL, input_id_format = "custom", verbose = FALSE)
merge_networks(list(first = first, second = second))
```

`mutations2ags` *Create altered gene sets from binary sample profiles*

Description

Convert a gene-by-sample mutation or other qualitative event matrix into one altered gene set (AGS) per sample. Every nonzero, nonmissing entry is treated as an event; event categories and magnitudes are otherwise ignored. Optional permutation preserves sample set sizes while randomly sampling genes according to their cohort-wide event frequencies.

Usage

```
mutations2ags(
  x,
  sample_columns = NULL,
  column_pattern = NULL,
  permute = FALSE,
  case_sensitive = FALSE,
  species,
  input_id_format = master_id_format(species),
  id_mappings = NULL,
  ambiguous = c("error", "drop", "first", "expand"),
  unmapped = c("keep", "drop", "error"),
  duplicate_ids = c("error", "mean", "sum", "max")
)
```

Arguments

`x`	Gene-by-sample matrix. Nonzero, nonmissing entries are altered.
`sample_columns`	Optional names or indices selecting sample columns.
`column_pattern`	Optional regular expression selecting sample columns.
`permute`	Randomize sets while preserving their sizes and accounting for relative event frequencies across the cohort.
`case_sensitive`	Use case-sensitive identifier matching. The default treats all gene identifiers case-insensitively.
`species`	Required species identifier accepted by `available_id_formats()`. Use explicit `NULL` for unsupported species or non-biological data.
`input_id_format`	Submitted identifier format. Defaults to `master_id_format()` for `species`; required when `species = NULL`.
`id_mappings`	Optional object from `load_id_mappings()`.
`ambiguous, unmapped`	Mapping policies passed to `map_identifiers()`.
`duplicate_ids`	Policy for rows that collapse to one identifier after mapping: error, mean, sum, or maximum.

Value

A named list containing one altered gene vector per sample, with normalization metadata available through `normalization_info()`.

See Also

`samples2ags()`, `import_gene_sets()`, `save_gs_list()`

Examples

```
events <- matrix(c(1, 0, 1, 0, 0, 1), nrow = 3,
  dimnames = list(c("TP53", "KRAS", "EGFR"),
    c("sample_1", "sample_2")))
mutations2ags(events, species = "human")
```

---

ndex_network_summary *Retrieve one NDEx network summary*

Description

Retrieve current metadata before deciding whether to download a network. This is an explicit online operation and returns the same columns as `search_ndex_networks()`.

Usage

```
ndex_network_summary(
  resource,
  server = NULL,
  access_key = NULL,
  username = NULL,
  password = NULL,
  token = NULL,
  timeout = 300
)
```

Arguments

`resource`	One NDEx network UUID, NDEx network URL, or one row returned by `search_ndex_networks()`.
`server`	NDEx server root or REST root. Defaults to the public NDEx server. Redirects are not followed. Internet-facing callers must allow-list trusted HTTPS origins rather than passing an arbitrary user-supplied URL.
`access_key`	Optional NDEx access key for an unlisted resource. It is used only for the request and is never written to a manifest.
`username, password`	Optional Basic authentication pair.
`token`	Optional bearer token. Do not combine with Basic credentials.
`timeout`	Positive request timeout in seconds.

Value

A one-row `nearender_ndex_catalog` data frame.

Examples

```
## Not run:
ndex_network_summary("f71ab602-97f0-11eb-9e72-0ac135e8bacf")

## End(Not run)
```

`nea.render` *Render altered gene sets as pathway-level network enrichment features*

Description

Given altered gene sets (AGSs), functional gene sets (FGSs), and a global network, evaluate enrichment from the number of network edges connecting each AGS–FGS pair. Expected counts are calibrated from the degree sums of all member nodes under the null model. Edges confined within only the AGS or only the FGS do not contribute. Returned matrices have FGSs in rows and AGSs in columns. Observed and expected edge counts and two-sided chi-square p-values accompany the signed z-score used for downstream modelling. Q-values follow the enrichment convention: BH correction is applied to all two-sided p-values, after which depleted scores are assigned q-value 1.

Usage

```
nea.render(
  ags,
  fgs,
  net,
  case_sensitive = FALSE,
  na_replace = 0,
  verbose = TRUE,
  network_name = NULL,
  members = FALSE,
  n_cores = 1L
)
```

Arguments

`ags`	Named list of altered gene sets.
`fgs`	Named list of functional gene sets.
`net`	Undirected igraph network, or a legacy network with a named adjacency list in `links` and an edge count in `Ntotal`.
`case_sensitive`	Use case-sensitive identifier matching. By default, identifiers in AGS, FGS, and NET are matched case-insensitively.

na_replace	Value used for zero observed link counts.
verbose	Report progress with message().
network_name	Optional non-empty network name recorded in the result. When omitted, a name stored on net is used, if available.
members	Whether to return the potentially large members.ags and members.fgs matrices of connected identifiers.
n_cores	Positive integer number of worker processes used only to format member matrices when members = TRUE. Sparse matrix calculations are not distributed across workers.

Value

A nearender_result. scores$z is the signed standardized deviation of observed AGS–FGS edge counts from the degree-based null. statistics contains AGS and FGS cumulative degrees, observed (n.actual) and expected (n.expected) edge counts, chi-square statistics, two-sided p-values, and BH-adjusted enrichment q-values (depleted cells are set to 1). When members = TRUE, it also contains connected AGS and FGS member matrices. The master z matrix occurs identically in both scores and statistics; all other matrices occur only in statistics.

References

Alexeyenko A et al. (2012). doi:10.1186/1471210513226. Jeggari A and Alexeyenko A (2017). doi:10.1186/s128590171534y.

See Also

ora.render(), ear(), rwr.render(), pagerank.render(), texture.render(), spectral.render(), gsva.render(), samples2ags(), mutations2ags(), set.heat()

Examples

```
data("NEArender.test")
ags <- NEArender.test$AGS$CCLE[1:2]
fgs <- NEArender.test$FGS$Hallmarks[1:3]
net <- NEArender.test$NET$STRING.300K
nea <- nea.render(ags, fgs, net, verbose = FALSE)
nea$scores$z
nea$statistics$q
```

NEArender.test *Real-world data for NEArender examples*

Description

AGS$CCLE: RNA-seq-based top-100 gene sets from 10 selected CCLE cancer cell lines. The CCLE values are batch-corrected log2(TPM + 1) expression values standardized per gene across all 1,673 models in DepMap 24Q4. AGS$LUAD: top-200 gene sets from 10 selected TCGA lung adenocarcinoma samples, using cBioPortal PanCancer Atlas log-transformed RNA Seq V2 RSEM z-scores relative to all samples. FGS$Hallmarks contains 50 gene sets from MSigDB Hallmarks release 6.0. mtr$CCLE and mtr$LUAD contain RNA-seq expression z-scores for the same CCLE cell lines and LUAD samples, respectively. panel$Olink.Inflammation250 contains the CCLE expression rows corresponding to the Olink.Inflammation250 gene panel. NET$STRING.300K contains the 300,000 highest-scoring unique edges from STRING version 12. Source edge scores were used for ranking but are not included as transition probabilities in analyses such as network flow. All gene identifiers use HGNC gene symbols.

Usage

```
data("NEArender.test")
```

Format

A nested list with FGS, AGS, NET, mtr, and panel components. The matrices have genes in rows and samples in columns.

Source

CCLE/DepMap: OmicsExpressionProteinCodingGenesTPMLogp1BatchCorrected.csv and Model.csv from DepMap 24Q4 Public, licensed under CC BY 4.0, doi:10.25452/figshare.plus.27993248.v1. TCGA-LUAD PanCancer Atlas 2018, cBioPortal DataHub study luad_tcga_pan_can_atlas_2018, RNA Seq V2 RSEM z-scores relative to all samples: https://github.com/cBioPortal/datahub/tree/master/public/luad_tcga_pan_can_atlas_2018. Hallmarks: MSigDB release 6.0, file h.all.v6.0.symbols.gmt, used under CC BY 4.0; doi:10.1016/j.cels.2015.12.004 and https://www.gsea-msigdb.org/gsea/msigdb. STRING: doi:10.1093/nar/gkac1000.

---

nearender_result *Renderer results in a uniform format*

Description

Every renderer returns the same list-based result class. Score matrices have functional gene sets in rows and samples or altered gene sets in columns. Every matrix identified by score_info$score_id occurs identically in both scores and statistics. Additional method-specific outputs occur only in statistics; statistical matrices are never duplicated at the top level.

Format

A list with components:

scores Named list of numeric score matrices.

score_info One metadata row per matrix in `scores`.

statistics The named score matrices plus method-specific statistics.

inputs Names and sizes of rendered inputs.

diagnostics Mapping and numerical diagnostics.

parameters Renderer parameters.

`nea_member_recovery_benchmark`

Benchmark a network using known gene sets

Description

Test a network's ability to recover known members of pathways or other functional sets by network enrichment. For each member, the function scores connectivity to the remaining members of the set to which it belongs. Randomly selected network nodes, optionally matched by node degree, provide negative controls for classical ROC or precision-recall evaluation. Test sets can be selected by biological context with `mask`. Curves for multiple networks can be combined by passing their returned predictions to `roc()`.

Usage

```
nea_member_recovery_benchmark(
  net,
  gs,
  mask = ".",
  min_n = 0,
  match_by_degree = TRUE,
  show_below_p = 1,
  mark_p = 0.01,
  mark_fdr = 0.1,
  graph = FALSE,
  seed = NULL,
  case_sensitive = FALSE,
  verbose = TRUE
)

benchmark(...)
```

Arguments

`net`	Undirected igraph network, sparse network, or legacy network with a named adjacency list in `links` and edge count in `Ntotal`.
`gs`	Named list of test gene sets.
`mask`	Regular expression selecting a subset of test-set names.
`min_n`	Minimum observed links required for a positive member.

match_by_degree
: Draw controls from the same rounded log2-degree bin; known members of every supplied gene set are excluded from the control pool. This is the recommended option.

show_below_p
: One scalar maximum one-sided enrichment p-value shown when graph = TRUE. The default, 1, retains the complete, canonical ROC curve.

mark_p
: Optional one scalar one-sided enrichment p-value marked with a circle. Use NULL to omit this marker.

mark_fdr
: Optional one scalar enrichment-only FDR threshold marked with a cross. Only positive enrichment scores are eligible for this marker.

graph
: Draw the curve.

seed
: Optional random seed for reproducible matched controls.

case_sensitive
: Use case-sensitive identifiers.

verbose
: Report progress with message().

...
: Arguments passed by the deprecated benchmark() alias.

Value

A prediction list accepted by roc(). tp, fp, tn, and fn contain cumulative classification counts at every value in cutoffs. circle_z and cross_z locate the p- and FDR-threshold markers; mark_p and mark_fdr retain their displayed probability values. nv and ne contain network vertex and edge counts.

References

Merid SK, Goranskaya D, Alexeyenko A (2014). Distinguishing between driver and passenger mutations in individual cancer genomes by network enrichment analysis. doi:10.1186/14712105-15308.

See Also

roc(), nea.render()

Examples

```
net <- list(links = list(
  a = c("b", "c"), b = c("a", "c"), c = c("a", "b", "d"),
  d = c("c", "e"), e = "d"), Ntotal = 5)
fgs <- list(pathway_1 = c("a", "b"), pathway_2 = c("d", "e"))
prediction <- nea_member_recovery_benchmark(net, fgs,
  show_below_p = 0.05, seed = 1, verbose = FALSE)
names(prediction)
```

network_catalog *List author-released network resources*

Description

This catalog complements `available_databases()` with network releases used for comparative benchmarking, including registered NDEx snapshots.

Usage

```
network_catalog()
```

Value

A data frame with one row per resource.

See Also

`download_resource()`, `prepare_resource()`

network_overlap *Calculate and present shared vertices and edges among networks*

Description

`network_overlap()` treats edges as undirected, collapses reciprocal and duplicate adjacency-list entries, and compares the resulting exact edge and vertex sets. Curated, experimental, and computational networks may have low exact overlap because each contains false-positive edges and misses real interactions. Overlap is therefore a structural comparison, not an absolute performance measure; use `nea_member_recovery_benchmark()` for pathway-member recovery. The function draws Venn diagrams for up to five networks by default and switches to UpSet-style plots for larger comparisons. Venn panels use expanded side margins and adaptive text for long names. Network labels include the corresponding total numbers of vertices or edges.

Usage

```
network_overlap(
  networks,
  plot = TRUE,
  plot_type = c("auto", "venn", "upset", "both"),
  max_intersections = 30L,
  case_sensitive = FALSE,
  colors = "style",
  opacity = 0.3,
  normalize = FALSE
)
```

Arguments

networks	Named list containing two or more networks. Each network may be an undirected igraph, a legacy named adjacency list with links and Ntotal, or an indexed network created internally by NEArender.
plot	Draw the vertex and edge overlap plots.
plot_type	One of "auto", "venn", "upset", or "both". Automatic mode uses Venn diagrams for at most five networks and UpSet-style plots for more. "both" draws Venn diagrams on the first page and UpSet-style plots on the second. Venn output supports at most seven networks.
max_intersections	Maximum number of observed intersections shown in each UpSet-style plot. All observed intersections are still returned as tables.
case_sensitive	Identifier case policy used when normalize = TRUE.
colors	Colors passed to venn::venn() as its zcolor argument.
opacity	Venn region opacity between zero and one.
normalize	Re-normalize network node identifiers. The default is FALSE because networks are normally normalized during import. Enabling this option warns because it copies and processes the full adjacency data.

Value

Invisibly, a list with per-network summaries and exact vertex and edge sets, intersection tables, and intersection members. In an intersection pattern, positions correspond to names(networks) and 1 denotes membership.

See Also

nea_member_recovery_benchmark(), node_degrees(), topology_2nd()

Examples

```
first <- list(links = list(a = "b", b = c("a", "c"), c = "b"),
  Ntotal = 2)
second <- list(links = list(b = "c", c = c("b", "d"), d = "c"),
  Ntotal = 2)
overlap <- network_overlap(list(first = first, second = second), plot = FALSE)
overlap$vertices$table
```

---

network_qc *Summarize network coverage and topology*

Description

Summarize network coverage and topology

Usage

```
network_qc(g, universe = NULL)
```

Arguments

g	A network accepted by as_igraph_network().
universe	Optional complete gene universe; absent genes receive degree zero.

Value

A one-row network quality-control data frame.

---

node_degrees *Investigate the degree distribution of a network graph*

Description

Plot the node-degree distribution and fit its log-log slope, for example as a diagnostic for assessing whether a network has a scale-free-like degree distribution. The returned exponent is the negative fitted slope.

Usage

```
node_degrees(
  net = NULL,
  main = "Node degree distribution",
  plot = TRUE,
  legend = TRUE,
  case_sensitive = FALSE,
  normalize = FALSE,
  nea = NULL,
  plot_type = c("points", "smooth")
)

connectivity(...)
```

Arguments

net	Undirected igraph network, or a legacy network with a named adjacency list in links and edge count in Ntotal.
main	Plot title.
plot	Draw the plot.
legend	Show network summary information.
case_sensitive	Identifier case policy used when normalize = TRUE.

normalize	Re-normalize network node identifiers. The default is FALSE because networks are normally normalized during import. Enabling this option warns because it copies and processes the full adjacency data.
nea	Deprecated compatibility alias for net.
plot_type	Plot the observed degree frequencies as ordinary points or as a smoothed density raster using graphics::smoothScatter(). The smoothed plot requires the suggested KernSmooth package.
...	Arguments passed by the deprecated connectivity() alias.

Value

Invisibly, the fitted exponent, with model and data attributes.

References

Clauset A, Shalizi CR, Newman MEJ (2009). Power-law distributions in empirical data. doi:10.1137/070710111.

See Also

topology_2nd(), network_overlap(), nea_member_recovery_benchmark()

Examples

```
data("NEArender.test")
net <- NEArender.test$NET$STRING.300K
exponent <- node_degrees(net, plot = FALSE)
attr(exponent, "data")
```

---

normalization_info *Inspect identifier-normalization metadata*

Description

Imported and prepared datasets record whether identifiers have already been normalized, which case policy was used, and an identifiers sublist with the declared species, submitted namespace, resulting namespace, and mapping provenance when applicable. Legacy unmarked objects return NULL.

Usage

```
normalization_info(x)
```

Arguments

x	A gene-set collection or network.

Value

A normalization metadata list, or NULL.

Examples

```
sets <- import_gene_sets(data.frame(gene = c("TP53", "MDM2")),
  layout = "single_set", species = "human")
normalization_info(sets)$identifiers
```

---

ontology_gene_set_api *Design of Disease Ontology and Gene Ontology gene-set services*

Description

Design record for constructing compact, versioned functional gene-set collections from the Human Disease Ontology (DO) and Gene Ontology (GO) for network enrichment analysis (NEA). This topic documents the source APIs and release files, ontology traversal rules, association evidence filters, species coverage, size rules, provenance requirements, and proposed service endpoints. NEArender implements the release-backed R adapter described here; the HTTP routes remain a service-layer design for a future website adapter.

Details

The common architecture is release-backed rather than a row-by-row proxy of a public ontology API. Public APIs remain useful for interactive term search and metadata lookup. Bulk gene-set construction uses pinned ontology and association artifacts, validates and caches them, maps identifiers into the NEArender master namespace, constructs the selected ontology closure, and serves immutable derived collections from the local cache.

Analytical functions remain offline. A download operation must be explicit, and a loaded collection must retain the provider, release, URLs, checksums, identifier mapping report, evidence policy, propagation relations, structural cut, member-count policy, and final dimensions in its manifest.

Disease Ontology sources and API feasibility

The DO knowledgebase provides an OpenAPI 3.1 REST API with endpoints for API metadata, all terms, term lookup by identifier or label, term search, imported ontologies, and prefixes. It is appropriate for discovery and term metadata, but it is not the source of human gene-disease associations.

For reproducible graph construction, use a dated official DO release. The primary `doid.owl` file is post-reasoned and contains more disease relationships and logical definitions. `doid.json` and `doid.obo` are alternative serializations, but OBO does not preserve logical definitions. The simplified `doid-base` and `doid-non-classified` products have fewer, pre-reasoned relations and must not be silently substituted for the primary file. Record both the release date and the selected release product.

Useful official resources are:

- https://disease-ontology.org/do-kb/api_doc/

- https://github.com/DiseaseOntology/HumanDiseaseOntology/blob/main/src/ontology/releases/README-release-files.md
- https://github.com/DiseaseOntology/HumanDiseaseOntology/releases

The between-term relations used for hierarchy calculations come from these official ontology release files. They do not come from repeated disease names or gene-disease rows in the DISEASES tab-separated files. The standard compact gene-set profile traverses disease `is_a` parentage. If inferred OWL relationships are used, the manifest must identify the reasoned release file and relationship set.

Disease-gene association files and confidence scores

The historical files inspected during this evaluation are under '`/home/proj/func/DATA/DO/2024`'. They are useful for local benchmarking and fast prototyping, but current acquisition must be based on provider documentation and a pinned release. These files are products of the DISEASES resource, whose disease identifiers are mapped to DO and ICD terms; they are not ontology graph files.

The four local files are approximately 613 MB for integrated associations, 6.3 MB for knowledge, 24.4 MB for experiments, and 46.1 MB for filtered text mining. Their modification dates alone are not sufficient release metadata. The ontology used for the reproducibility check was the immediately preceding official `2024-03-28` DO release.

All four formats start with gene identifier, gene name, disease identifier, and disease name. Their remaining columns are:

'`human_disease_integrated_full.tsv`' The fifth and final column is the integrated confidence score.

'`human_disease_knowledge_full.tsv`' Source database, evidence type, and confidence score. The confidence score is column seven.

'`human_disease_experiments_full.tsv`' Source database, source score, and confidence score. The confidence score is column seven; the source score and normalized confidence must not be confused.

'`human_disease_textmining_filtered.tsv`' Text-mining z-score, confidence score, and evidence-viewer URL. The confidence is column six, not the final URL column.

DISEASES documents its confidence as a harmonized one-to-five-star scale derived differently for curated knowledge, experimental/GWAS, and text-mined evidence. The inspected 2024 integrated file contains values from 0.5 through 5.0. Consequently, association counts must never treat every row as equally supported. Apply a declared channel and minimum confidence to direct input rows before gene-term deduplication or ontology propagation. Preserve the original channel, source score, confidence, and threshold in provenance.

The DISEASES 2.0 documentation assigns direct curated knowledge high evidence, derives text-mining confidence from its z-score, and calibrates experimental TIGA evidence against a benchmark. Its published overview uses confidence at least 3. The resource also backtracks associations to broader ontology parents. The inspected files visibly contain those ancestor rows. Therefore, the standard historical-data adapter filters and deduplicates the existing integrated rows but does not propagate them a second time. Propagation is enabled only for an explicitly direct association product. See doi:10.1093/database/baac019.

The integrated channel is the convenient default when one comparable score is needed. Channel-specific profiles should remain available for users who want, for example, curated knowledge without text mining. When multiple retained rows support the same mapped gene-term pair, retain provenance for all rows and use a documented aggregation rule, such as the maximum integrated confidence, rather than summing evidence that may be redundant.

Provider format documentation is available at https://diseases.jensenlab.org/downloads.

Disease hierarchy cut for NEA

A fixed ontology depth is unsuitable because DO branches have unequal depth and multiple parentage. The inspected 2024 graph contained 11,537 active terms, 9,160 active terminal terms, and 16,058 `is_a` edges. Construct an adaptive structural frontier instead:

1. Remove obsolete terms and non-DO pseudo-terms, such as ICD hierarchy nodes, from the standard candidate collection.
2. Traverse the selected official DO parent relations and count the terminal descendants of each disease term.
3. For term t, calculate structural information content `IC_s(t) = -log2(L(t) / L(root))`, where `L(t)` is its number of terminal descendants.
4. Select the topmost terms that cross the structural-specificity threshold. Selecting every term below the threshold would retain most of the ontology and would not produce a compact collection.
5. Intersect mapped members with the analysis network universe, require at least three genes, and prune nearly identical ancestor-descendant sets. A starting redundancy rule is Jaccard similarity of at least 0.8, retaining the more specific term.

The agreed historical preset is:

```
association_file = human_disease_integrated_full.tsv
ontology_release = 2024-03-28
min_confidence = 3
max_leaf_descendants = 100
min_genes = 3
```

With 9,160 root terminal descendants, the 100-leaf cap is approximately `IC_s = 6.5`. The preset produces 1,538 terms. It removes very broad terms without relying on a manually maintained blacklist: “genetic disease” has 4,238 terminal descendants, “disease of cellular proliferation” has 1,955, and “cell type cancer” has 696.

The minimum member count is three. The conventional GSEA lower limit of 15 is not applicable to NEA because NEA evaluates network edges rather than only set-member overlap and can detect signal from very small sets.

Do not impose a universal upper limit of 200. Under the agreed preset, the maximum observed membership is 319 genes, tied by “autoimmune disease of musculoskeletal system” and “artery disease”. Ovarian cancer has 55 terminal descendants and 80 genes at confidence at least 3; type 2 diabetes mellitus has one terminal descendant and 124 genes. At lower confidence thresholds both diseases become much larger, demonstrating why confidence filtering must precede member-count assessment.

A structurally specific, 319-member disease should not be removed merely because it is well studied. Expose an optional `max_genes` parameter, but make the standard decision from ontology specificity and annotation information content: `IC_a(t,s) = -log2(|G(t,s) intersect U(s)| / |G(root,s) intersect U(s)|)`. This makes the broadness rule adaptive to the organism and network universe.

Gene Ontology source architecture

GO is suitable for a NEArender service covering all supported organisms. Use `go-basic.obo` or `go-basic.json` for the ontology and one official organism-specific GAF file per species. Companion GPI files should be retained where they improve identifier and cross-reference mapping. The inspected GAF products total approximately 46 MB compressed and `go-basic.obo` is approximately 32 MB, making an all-species refresh practical.

`go-basic` is acyclic and intended for annotation propagation. The standard profile traverses only `is_a` and `part_of`. Both are safe for grouping annotations. `has_part` is not safe in that direction, and `regulates`, `positively_regulates`, and `negatively_regulates` change the biological meaning of membership. Regulation edges may be offered as a separately labelled causal profile but must not be silently included in the standard closure.

Official resources are:

- https://geneontology.org/docs/download-ontology/
- https://geneontology.org/docs/download-go-annotations/
- https://geneontology.org/docs/download-go-annotations/downloads/
- https://geneontology.org/docs/ontology-relations/
- https://geneontology.org/docs/tools-guide/

The live GO API can support interactive term lookup. Reproducible bulk collections must be built from release files rather than mining the shared API one association at a time.

GO evidence policy

GO evidence codes describe provenance and method; they are not an ordinal confidence score. Expose both named profiles and exact include/exclude lists:

`all_positive` All positive evidence, including IEA, after removing NOT and ND annotations.

`non_iea` All positive evidence except IEA, after removing ND. This is the recommended default. It must not be labelled "high confidence", because the remaining categories are heterogeneous.

`experimental` EXP, IDA, IPI, IMP, IGI, IEP, HTP, HDA, HMP, HGI, and HEP.

The complete category mapping used by the adapter is:

Experimental	EXP, IDA, IPI, IMP, IGI, IEP, HTP, HDA, HMP, HGI, HEP
Phylogenetic	IBA, IBD, IKR, IRD
Computational analysis	ISS, ISO, ISA, ISM, IGC, RCA
Author statement	TAS, NAS
Curator statement	IC, ND
Automatic	IEA

For positive NEA gene sets, discard GAF rows whose relation/qualifier contains `NOT`, and discard `ND`, which records that no biological data were found. Retain negative annotations separately if they are needed for auditing; never convert them to positive membership. Apply the evidence filter to direct annotations before ancestor propagation. Otherwise an excluded IEA annotation would reappear as inherited membership. After filtering, propagate over the selected safe relations and deduplicate gene-term pairs. Preserve the exact supporting codes and whether membership is direct or inherited.

The relevant GAF columns are relation/negation in column four, GO identifier in column five, evidence code in column seven, aspect in column nine, and taxon in column 13. See https://geneontology.org/docs/guide-go-evidence-codes/ and https://geneontology.org/docs/go-annotation-file-gaf-format-2.2/.

Compact GO frontiers

GO has three roots and must be treated as three separate DAGs. In the inspected `go-basic` release, after removing obsolete terms and retaining only `is_a` and `part_of`, the graph contained:

Aspect	Active terms	Terminal terms
Biological Process	23,974	11,858
Molecular Function	10,041	8,065
Cellular Component	4,077	2,796

Calculate structural information content separately within each aspect: `IC_s(t) = -log2(L(t) / L(aspect_root))`. Again, select only the topmost crossing frontier. Merely retaining all terms above a specificity threshold keeps nearly the whole ontology because most GO terms are terminal or close to terminal.

Empirical topmost-frontier sizes were:

Structural cutoff	BP	MF	CC	Total
IC_s at least 5.0	796	1,191	833	2,820
IC_s at least 6.5	1,748	2,598	1,246	5,592

Use `IC_s = 5.0` as an initial compact profile and `IC_s = 6.5` as a more specific, larger profile. Then apply species/network-specific annotation information content, require at least three mapped genes, and prune near-duplicate ancestor-descendant sets. A starting annotation cutoff of `IC_a = 3` excludes sets occupying more than one eighth of the annotated network universe.

Exclude roots, obsolete terms, and terms in the `gocheck_do_not_annotate` subset from candidate output, but do not remove their descendants. The inspected release contained 459 such terms. The generic GO slim contained only about 140 active terms and is useful for summaries or coverage checks, but is too coarse to be the principal NEA enrichment collection. See https://geneontology.org/docs/go-subset-guide/.

As for DO, 319 genes is not a universal exclusion boundary. A structurally precise complex or process may legitimately have more than 200 members. Leave `max_genes` unset in standard profiles and use annotation information content; retain an explicit maximum only as a caller-selected option.

GO coverage of NEArender organisms

The official organism products cover all eight species in the NEArender identifier catalog. Counts below are raw positive GAF object identifiers after the stated evidence filter, before ontology propagation, NEArender identifier conversion, or network-universe intersection.

NEArender code	Official GAF product	Non-IEA genes	Experimental genes
hsa	HUMAN-uniprot.gaf.gz	22,029	18,343
mmu	MOUSE-mod.gaf.gz	20,430	13,698
rno	RAT-mod.gaf.gz	18,821	6,471
dme	DROME-mod.gaf.gz	11,852	7,269
cel	CAEEL-mod.gaf.gz	9,964	4,303
sce	YEAST-mod.gaf.gz	6,676	6,202
ath	ARATH-mod.gaf.gz	25,834	14,779
eco	ECOLI-mod.gaf.gz	4,063	3,658

Human annotations require UniProt-to-HGNC conversion. The mouse, rat, fly, worm, yeast, and Arabidopsis MOD identifiers align with corresponding mapping catalog namespaces, although they must still pass through the normal mapping cache. E. coli EcoCyc identifiers require GPI/xref mapping or a documented gene-symbol fallback.

Taxon validation must accept documented strain mappings rather than require literal equality. The yeast GAF uses S288C taxon 559292 while the NEArender catalog represents species taxon 4932. The E. coli GAF uses K-12 taxon 83333 while the package profile represents MG1655 taxon 511145.

Implemented R interface and future service routes

The package interface separates source acquisition from offline parsing:

```
available_ontology_collections()
download_ontology_collection("go_hsa")
go <- load_ontology_collection("go_hsa")

do <- import_do_collection(
    ontology = "doid.obo",
    integrated = "human_disease_integrated_full.tsv"
)
# Optional cached channels are parsed only when requested:
do_all_scores <- load_ontology_collection("DO", channels = "all")
```

Each loaded collection has five synchronized components:

`fgs` An ordinary named term-to-gene list keyed as `term_id;term_name` and accepted directly by NEArender functions.

`associations` GO terms contain named character vectors whose values are sorted semicolon-delimited evidence codes. DO terms contain named numeric vectors of integrated scores.

`membership` A compact flat term–gene table containing separate term identifiers and names, canonical gene identifiers, and relevant evidence or score columns.

`terms` The complete term-feature table with graph depths, terminal descendants, structural and annotation information content, member counts, evidence fractions, and summaries for loaded score channels.

`metadata` Interpretation metadata: GO evidence-code explanations and policies, or the complete DO scoring-system description including unloaded channels.

List keys combine the stable identifier and label as `term_id;term_name`; the separate table columns remain authoritative for programmatic filtering. Provider evidence codes and confidence scores retain source precision; derived summaries are rounded to three decimal places. A DAG has no unique ontology level, so both minimum and maximum root depth are exposed.

For a website, the same query model can serve both ontologies while retaining source-specific controls. Acquisition and parsing should run as background jobs against immutable snapshots; interactive requests should read a prepared snapshot or database projection rather than contact upstream providers.

A common query model can serve both ontologies while retaining source-specific controls. Example GO request:

```
GET /go/gene-sets
    ?species=hsa
    &aspect=BP
    &evidence_profile=non_iea
    &relations=is_a,part_of
    &structural_ic=5
    &annotation_ic=3
    &min_genes=3
    &release=YYYY-MM-DD
```

Example disease request:

```
GET /do/gene-sets
    ?species=hsa
    &association_profile=integrated
    &min_confidence=3
    &relations=is_a
    &max_leaf_descendants=100
    &min_genes=3
    &release=YYYY-MM-DD
```

The disease confidence threshold and leaf cap reproduce the agreed historical preset. They must be declared in the collection manifest and are not interchangeable with GO evidence categories.

Supporting endpoints may include:

```
GET /ontologies/releases
GET /ontologies/species
GET /do/terms/{doid}
GET /go/terms/{go_id}
GET /do/gene-sets/{doid}
GET /go/gene-sets/{species}/{go_id}
GET /ontologies/download/{collection_id}
```

Each returned gene set should include term identifier, label, ontology/aspect, mapped members, direct and inherited counts where available, evidence or confidence provenance, structural and annotation information content, terminal-descendant count, identifier-mapping losses, network-universe size, release identifiers, source URLs, checksums, and license/attribution fields.

Release and reproducibility requirements

Never assume that files behind independent "current" URLs were built against exactly the same ontology version. Read and store ontology versions from headers, prefer products from a single dated release bundle, and warn or fail on incompatible combinations. GO GAF headers expose their ontology version; DO OWL/JSON/OBO files expose a version IRI or data version. A derived collection identifier must include the ontology release, annotation release, evidence/confidence profile, relationship closure, structural and annotation cuts, identifier-mapping release, and parser version.

GO release and citation guidance is at https://geneontology.org/docs/go-citation-policy/. Both the GO and DISEASES products require their stated attribution when derived collections are redistributed. Provider terms and licenses must be copied into the cached manifest rather than inferred from file extensions or remembered defaults.

See Also

available_ontology_collections(), download_ontology_collection(), load_ontology_collection(), import_go_collection(), import_do_collection(), filter_ontology_collection(), available_databases(), download_geneset_database(), load_geneset_database(), available_id_formats(), prepare_id_mappings(), manifest()

---

ora.render *Render altered gene sets by over-representation analysis*

Description

Render altered gene sets by over-representation analysis

Usage

```
ora.render(
  ags,
  fgs,
  universe,
  case_sensitive = FALSE,
  n_cores = 1L,
  verbose = TRUE
)
```

Arguments

ags	Named list of altered gene sets.
fgs	Named list of functional gene sets.
universe	Character vector of eligible gene or protein identifiers, or a positive integer giving only the universe size. An explicit vector is preferred.
case_sensitive	Use case-sensitive identifier matching. By default, identifiers in AGS, FGS, and the universe are matched case-insensitively.
n_cores	Positive integer number of worker processes. Independent FGS rows are distributed across workers when greater than one.
verbose	Report progress with `message()`.

Details

Each AGS–FGS pair is tested with a two-sided Fisher exact test. BH adjustment is applied across all cells, after which depleted pairs (odds ratio below 1) receive q-value 1 to retain the enrichment-only convention used by NEArender.

Value

A `nearender_result`. `scores$odds_ratio` contains Fisher-test odds ratios. The identical master matrix is also available as `statistics$odds_ratio`; other statistics are literal overlap counts (`n.actual`), two-sided p-values, and enrichment q-values. No statistical matrices occur at the result's top level. `inputs` records the analyzed set names, sizes, and universe size.

See Also

`nea.render()`, `ear()`, `gsva.render()`

Examples

```
data("NEArender.test")
ags <- NEArender.test$AGS$CCLE[1:2]
fgs <- NEArender.test$FGS$Hallmarks[1:3]
universe <- unique(c(unlist(ags), unlist(fgs)))
ora <- ora.render(ags, fgs, universe, verbose = FALSE)
ora$scores$odds_ratio
ora$statistics$p
```

---

pagerank.render *Render gene sets by personalized PageRank*

Description

Nodes in each AGS supply a uniform personalization vector to `igraph::page_rank()`. PageRank probabilities are summed over each FGS.

Usage

```
pagerank.render(
  ags,
  fgs,
  net,
  damping = 0.15,
  use_edge_weights = FALSE,
  case_sensitive = FALSE,
  n_cores = 1L,
  verbose = TRUE
)
```

Arguments

ags	Named list of altered gene sets.
fgs	Named list of functional gene sets.
net	Undirected igraph network, or a legacy network with a named adjacency list in `links` and an edge count in `Ntotal`.
damping	PageRank damping factor in the interval `(0, 1)`. Increasing it raises the probability of following network edges, lowers the implicit restart probability `1 - damping`, and permits longer-range diffusion. Thus its direction is opposite to increasing `restart` in `rwr.render()`.
use_edge_weights	Use the igraph `weight` edge attribute to determine transition probabilities. The default is `FALSE` because imported edge weights may encode evidence or confidence rather than a transition model.
case_sensitive	Use case-sensitive identifier matching. By default, identifiers in AGS, FGS, and NET are matched case-insensitively.
n_cores	Positive integer number of worker processes. Independent AGS personalization vectors are distributed across workers when greater than one.
verbose	Report progress with `message()`.

Details

With `use_edge_weights = FALSE`, NEArender explicitly suppresses igraph's automatic use of a `weight` attribute. With `TRUE`, each source node's edge weights are normalized to transition probabilities by PageRank. Returned pathway affinities are relative scores, not p- or q-values.

Value

A `nearender_result`. `scores$pagerank` is an FGS-by-AGS matrix obtained by summing personalized PageRank probabilities over each FGS; `statistics$pagerank` is the identical master score matrix. `diagnostics` reports original and mapped AGS/FGS sizes, and `parameters` records damping, edge-weight use, case policy, and workers.

See Also

`rwr.render()`, `nea.render()`, `spectral.render()`

Examples

```
edges <- data.frame(from = c("A", "A", "B"), to = c("B", "C", "C"),
  confidence = c(5, 1, 2))
net <- import_network(edges, "from", "to", "confidence",
  species = NULL, input_id_format = "custom", verbose = FALSE)
ags <- list(sample = "a")
fgs <- list(pathway_b = "b", pathway_c = "c")
page <- pagerank.render(ags, fgs, net, verbose = FALSE)
page$scores$pagerank
```

`plot_subgraph` *Extract and plot one pathway subgraph*

Description

Quickly extract the network induced by one pathway and optionally draw it with igraph. Pathway members are retained even when they are isolated in the selected scope, and the extracted igraph object is returned invisibly.

Usage

```
plot_subgraph(
  net,
  fgs,
  scope = c("within-pathway", "boundary-expanded"),
  attributes = NULL,
  plot = TRUE,
  main = NULL,
  case_sensitive = FALSE,
  ...
)
```

Arguments

`net`	An undirected igraph network, a legacy network with a named adjacency list in `links`, an indexed sparse NEArender network, or a sparse adjacency matrix.
`fgs`	A character vector containing one pathway, or a one-element named gene-set list. A named list supplies the default plot title.
`scope`	Graph scope. `"within-pathway"` returns the subgraph induced by mapped pathway members. `"boundary-expanded"` also includes all one-hop neighbors and edges among the selected nodes.
`attributes`	Optional node display attributes. Supply a named vector to color-code nodes, a named list with `color` and/or `size` vectors, or a data frame or matrix with node identifiers as row names and `color` and/or `size` columns. Numeric color values use a continuous palette; categorical values use a discrete palette; valid R colors are used literally. Sizes are non-negative values in igraph plotting units. Attribute node names use the same case policy as `fgs` and `net`; omitted or `NA` values retain the defaults.

plot	Draw the extracted graph.
main	Plot title. By default, use the pathway name when fgs is a named list, otherwise "Pathway subgraph".
case_sensitive	Use case-sensitive identifier matching.
...	Additional arguments passed to igraph's plot method. Explicit vertex.color and vertex.size arguments override the stored display attributes for that drawing.

Details

By default, pathway members are orange and boundary nodes are blue. The returned graph has a logical pathway_member vertex attribute, display color (and, when supplied, size) vertex attributes, and retains source edge weights. Partial pathway and attribute mapping is allowed, but at least one pathway member must map to the network.

Value

Invisibly, a simple undirected igraph graph for the selected scope.

See Also

spectral_eigenpairs(), as_igraph_network()

Examples

```
net <- list(
  links = list(A = "B", B = c("A", "C"), C = "B"),
  Ntotal = 2)
graph <- plot_subgraph(net, list(signaling = c("A", "B")),
  scope = "boundary-expanded", plot = FALSE,
  attributes = list(color = c(A = "activated", B = "repressed"),
    size = c(A = 20, B = 12)))
igraph::vertex_attr(graph, "pathway_member")
```

---

prepare_id_mappings *Download and prepare identifier mappings*

Description

Explicitly acquire mapping data only for the requested species, transform it into a release-specific long table, and cache the result. BioMart attributes are queried independently to avoid Cartesian expansion across transcript and peptide identifiers. E. coli mappings supplement NCBI Gene symbols and locus tags with reviewed UniProtKB accessions. No renderer or package installation invokes this function. This is the explicit online first step in the public mapping workflow: call prepare_id_mappings(), then load_id_mappings(), then either pass the loaded object to an importer or call map_identifiers() directly.

Usage

```
prepare_id_mappings(
  species,
  release = "current",
  cache_dir = NULL,
  force = FALSE,
  quiet = FALSE
)
```

Arguments

species	One or more supported scientific names, common names, package species IDs, three-letter KEGG style organism IDs, or NCBI taxonomy IDs.
release	'current' to reuse the checksum-validated local current release when available, or otherwise discover and record the current provider release. A concrete release must already be cached.
cache_dir	Explicit cache root required for this writing operation.
force	Contact the provider even when a valid current cache exists, and replace a valid cached mapping when the resolved release is unchanged.
quiet	Suppress provider download progress.

Value

A named list containing cached artifact paths, invisibly.

See Also

load_id_mappings(), map_identifiers()

Examples

```
cache <- file.path(tempdir(), "NEArender-cache")
## Not run:
prepare_id_mappings("human", cache_dir = cache)
mappings <- load_id_mappings("human", cache_dir = cache)

## End(Not run)
```

---

prepare_resource *Download, harmonize, and import an author-released network*

Description

This high-level interface implements the resource workflow represented by network_catalog(). It never runs implicitly: callers must provide an explicit cache directory, and large resources require confirmation.

Usage

```
prepare_resource(
  id,
  cache_dir,
  mappings = NULL,
  n_top = NULL,
  allow_large = FALSE,
  paths = NULL,
  universe = NULL,
  mapper = NULL,
  chunk_size = 100000L,
  min_score = NULL,
  quiet = FALSE
)
```

Arguments

id	Resource identifier or specification.
cache_dir	Explicit cache root.
mappings	Optional identifier mappings from load_id_mappings().
n_top	Optional number of highest-scoring mapped edges.
allow_large	Confirm a resource marked as a large download.
paths	Optional named paths returned by download_resource().
universe	Optional vector restricting both mapped endpoints.
mapper	Optional endpoint mapping function, overriding mappings.
chunk_size	Positive number of source rows parsed per chunk.
min_score	Optional minimum author score.
quiet	Suppress download progress.

Value

An undirected igraph network with a provenance manifest available through manifest().

---

read_network *Read a third-party graph file with igraph*

Description

Read a third-party graph file with igraph

Usage

```
read_network(
  path,
  format = c("graphml", "gml", "ncol", "lgl", "pajek", "edgelist"),
  output_network_format = c("igraph", "sparse", "legacy"),
  case_sensitive = FALSE,
  ...,
  network_format = NULL,
  species,
  input_id_format = master_id_format(species),
  id_mappings = NULL,
  ambiguous = c("error", "drop", "first", "expand"),
  unmapped = c("keep", "drop", "error")
)
```

Arguments

path	Input graph file.
format	An igraph reader format.
output_network_format	Return an igraph, sparse, or legacy NEArender network.
case_sensitive	Preserve identifier case.
...	Additional arguments passed to igraph::read_graph().
network_format	Deprecated alias for output_network_format.
species	Required species identifier accepted by available_id_formats(). Use explicit NULL for unsupported species or non-biological data.
input_id_format	Submitted identifier format. Defaults to master_id_format() for species; required when species = NULL.
id_mappings	Optional object from load_id_mappings(). When supplied, both endpoints are converted to its species master format.
ambiguous, unmapped	Mapping policies passed to map_identifiers().

Value

A normalized igraph, sparse, or legacy NEArender network.

See Also

import_network(), export_network(), convert_network()

Examples

```
graph <- import_network(data.frame(from = "A", to = "B", score = 2),
  score_column = "score", species = NULL, input_id_format = "custom",
```

```
  verbose = FALSE)
path <- tempfile(fileext = ".graphml")
export_network(graph, path)
restored <- read_network(path, species = NULL, input_id_format = "custom")
unlink(path)
igraph::edge_attr(restored, "weight")
```

`reduce_network` *Reduce a scored network*

Description

Retain edges from a scored network using either a minimum score or a top-edge rule. Source edge weights and identifier metadata are preserved, and the returned object's `manifest()` records the filter and source provenance.

Usage

```
reduce_network(
  net,
  select = list(min_score = NULL, n_top = NULL, random = FALSE),
  greedy = TRUE,
  output_network_format = c("same", "igraph", "sparse", "legacy"),
  case_sensitive = NULL
)
```

Arguments

`net`	A supported weighted network object.
`select`	Named selection list with `min_score`, `n_top`, and `random`. Specify either `min_score`, or `n_top`. With `n_top`, set `random = TRUE` to sample that many unique edges instead of ranking them by score.
`greedy`	Boundary-tie policy used with `n_top`. The default, `TRUE`, includes all edges tied at the boundary and can exceed `n_top`. `FALSE` excludes the whole boundary tie and can return fewer edges. `NULL` returns exactly `n_top` edges, resolving a boundary tie by alphanumeric canonical-endpoint order. Strict `greedy = NULL` selection is useful for size-matched network benchmarks.
`output_network_format`	Output representation. `"same"` preserves the input representation.
`case_sensitive`	Identifier case policy. By default, reuse the recorded policy of `net`, or use case-insensitive matching for an unmarked object.

Value

A reduced network with a transformation provenance manifest.

See Also

merge_networks(), load_network_database(), manifest()

Examples

```
edges <- data.frame(from = c("A", "A", "B", "C"),
  to = c("B", "C", "C", "D"), score = c(5, 4, 4, 2))
network <- import_network(edges, score_column = "score", species = NULL,
  input_id_format = "custom", verbose = FALSE)
reduce_network(network, select = list(n_top = 2))
```

---

render_panel *Render a harmonized panel of author networks*

Description

Render a harmonized panel of author networks

Usage

```
render_panel(networks, ags, fgs, scope = c("comprehensive", "restricted"))
```

Arguments

networks	Named list of networks returned by prepare_resource().
ags, fgs	Altered and functional gene-set lists.
scope	Render comprehensive or restricted networks.

Value

A named list of nea.render() results.

---

roc *Plot ROC or precision-recall curves*

Description

Plot one or more network member-recovery benchmarks. ROC mode reports normalized specificity and sensitivity. max_p truncates curves at a threshold specified by a one-sided enrichment p-value. Precision-recall mode reports recall and precision as proportions, uses precision 1 before the first positive prediction, and shows the class prevalence as its no-skill baseline.

Usage

```
roc(
  tpvsfp,
  mode = "roc",
  max_p = 1,
  coff_z = NULL,
  cex_main = 1,
  cex_leg = 0.75,
  main = NULL,
  use_lty = FALSE,
  print_stats = TRUE,
  sort_by_letter_and_remove = FALSE
)
```

Arguments

tpvsfp	Prediction list from nea_member_recovery_benchmark(), or a named list of them.
mode	Plot coordinates. Supported values are "roc" (an alias for "specificity_vs_sensitivity"), "prc" (an alias for "sensitivity_vs_tdp"), "recall_vs_tdp", "f1_vs_tdp", "tdp_vs_mcc", "sensitivity_vs_f1", and "sensitivity_vs_mcc".
max_p	Maximum one-sided enrichment p-value shown in ROC mode. The default, 1, retains the complete curve.
coff_z	Deprecated direct cutoff for the NEA z-score. It remains available when max_p is not supplied; supplying both explicitly is an error.
cex_main, cex_leg	Text sizes for title and legend.
main	Plot title.
use_lty	Use varying line types. For more than six curves, colors cycle through a six-color palette before line types 1, 2, and 3 cycle.
print_stats	Show node and edge counts. Counts shared by all curves are printed once below the main title; differing counts appear in the legend.
sort_by_letter_and_remove	Remove a leading letter-and-colon prefix.

Details

Curves and their colors are ordered alphabetically by their displayed labels. A prediction element may contain scalar col and lty fields to override its automatically assigned style. Differing network sizes are appended on the same legend line for labels of at most 24 characters. Longer labels put the sizes on a second line and use slightly smaller legend text. Common sizes are printed once below the title, which is useful when curves compare renderers on the same network. When benchmark marker thresholds are available, a top-left legend identifies the circle as p and the cross as FDR.

OR (odds ratio), shown next to each network name in the legend when mark_fdr is not NULL and returned as odds_ratio, is calculated as sensitivity / (1 - specificity) (Merid et al., 2014), i.e. sensitivity divided by the false-positive rate at the cutoff selected by mark_fdr. Both sensitivity

and the false-positive rate must be non-zero; otherwise OR is NA. With equally sized positive and negative gene sets, a higher OR at a fixed mark_fdr indicates a lower false-positive rate for a given sensitivity. This is not the cross-product odds ratio of a 2 by 2 contingency table. The returned column contains the same ratio at every displayed cutoff so that the marked value can be inspected directly.

Value

Invisibly, an alphabetically named list of data frames containing the x coordinate, y coordinate, score cutoff, and Merid-style odds_ratio used for every displayed curve. ROC coordinates are specificity and sensitivity; precision-recall coordinates are recall and true discovery proportion.

References

Merid SK, Goranskaya D, Alexeyenko A (2014). doi:10.1186/1471210515308.

See Also

nea_member_recovery_benchmark()

Examples

```
net <- igraph::graph_from_edgelist(matrix(c(
  "b", "c", "b", "d", "e", "f", "a", "g", "f", "g",
  "b", "h", "d", "h", "c", "i", "a", "j", "b", "j",
  "e", "j", "h", "k", "c", "l"
), ncol = 2, byrow = TRUE), directed = FALSE)
prediction <- nea_member_recovery_benchmark(net,
  list(pathway = c("a", "b", "c", "d", "g")),
  seed = 7, verbose = FALSE)
curves <- roc(list(example_network = prediction), mode = "roc")
curves$example_network
```

---

run_renderer *Run a renderer through a common interface*

Description

run_renderer() selects a rendering method, validates which primary inputs are relevant to it, and forwards the named options in renderer_args. The selected renderer's nearender_result is returned unchanged.

Usage

```
run_renderer(
  method = c("nea", "ora", "rwr", "pagerank", "texture", "spectral", "gsva"),
  ags = NULL,
  fgs = NULL,
  net = NULL,
```

```
  mtr = NULL,
  universe = NULL,
  renderer_args = list()
)
```

Arguments

method	One renderer name: "nea", "ora", "rwr", "pagerank", "texture", "spectral", or "gsva".
ags	Named list of altered gene sets. Required by "nea", "ora", "rwr", and "pagerank".
fgs	Named list of functional gene sets. Required by every renderer.
net	Network accepted by the selected renderer. Required by "nea", "rwr", "pagerank", "texture", and "spectral".
mtr	Numeric gene-by-sample matrix. Required by "texture", "spectral", and "gsva".
universe	Gene identifiers defining the ORA universe, or one positive integer giving its size. Required only by "ora".
renderer_args	Named list of additional arguments forwarded to the selected renderer. Dispatcher-managed core inputs cannot be overridden. For the GSVA renderer, use renderer_args = list(method = "zscore"), for example, to set the method argument of gsva.render().

Details

The primary input requirements are:

method	Required primary inputs
"nea", "rwr", "pagerank"	ags, fgs, net
"ora"	ags, fgs, universe
"texture", "spectral"	fgs, net, mtr
"gsva"	fgs, mtr

Supplying a primary argument that is irrelevant to the requested renderer, even explicitly as NULL, is an error. The diagnostic describes the renderer and lists its relevant inputs so that, for example, network data cannot be silently ignored by ORA or GSVA-family scoring. Required arguments that were supplied but evaluated to NULL, commonly because a list name did not match, are distinguished from arguments that were omitted.

Value

A nearender_result produced by the selected renderer.

See Also

nea.render(), ora.render(), rwr.render(), pagerank.render(), texture.render(), spectral.render(), gsva.render()

Examples

```
ags <- list(sample = c("A", "B"))
fgs <- list(pathway = c("B", "C"))
result <- run_renderer(method = "ora", ags = ags, fgs = fgs,
  universe = c("A", "B", "C", "D"),
  renderer_args = list(verbose = FALSE))
result$scores$odds_ratio
```

`rwr.render` *Render gene sets by random walk with restart*

Description

Nodes in each AGS define a uniform restart distribution. Steady-state node affinities are summed over the members of each FGS. This sparse-matrix implementation follows the `dnet::dRWR()` algorithm without requiring the dnet package at run time.

Usage

```
rwr.render(
  ags,
  fgs,
  net,
  restart = 0.75,
  normalise = c("laplacian", "column", "row", "none"),
  tol = 1e-06,
  max_iter = 50L,
  use_edge_weights = FALSE,
  case_sensitive = FALSE,
  verbose = TRUE
)
```

Arguments

`ags`	Named list of altered gene sets.
`fgs`	Named list of functional gene sets.
`net`	Undirected igraph network, or a legacy network with a named adjacency list in `links` and an edge count in `Ntotal`.
`restart`	Restart probability in `[0, 1]`. Increasing it concentrates affinity nearer the AGS seeds and reduces contributions from longer paths.
`normalise`	Adjacency normalization method. The `"column"` and `"row"` labels describe the normalized matrix margin and differ from the dnet labels.
`tol`	Positive L1 convergence tolerance.
`max_iter`	Positive maximum iteration count.

use_edge_weights
: Use the igraph `weight` edge attribute to determine transition probabilities. The default is `FALSE` because imported edge weights may encode evidence or confidence rather than a transition model.

case_sensitive
: Use case-sensitive identifier matching. By default, identifiers in AGS, FGS, and NET are matched case-insensitively.

verbose
: Report progress with `message()`.

Details

Node-affinity vectors are stored as columns and updated by left multiplication with the transition matrix. Under this convention, `normalise = "column"` constructs `A D^-1`, whose columns sum to one, so each source node distributes its probability among its neighbors. `normalise = "row"` constructs `D^-1 A`, whose rows sum to one; when applied to column vectors, this is a different, non-probability-conserving propagation rule. For an undirected network the two matrices are transposes, but they coincide only when connected nodes have equal degrees. The `dnet::dRWR()` interface used the opposite labels; NEArender names the options for the margin of the resulting transition matrix that is actually normalized.

With `use_edge_weights = FALSE`, every retained edge has equal weight. With `TRUE`, the non-negative igraph `weight` attribute is normalized according to `normalise` to derive propagation coefficients; for `normalise = "column"`, these are transition probabilities. The returned affinities are relative scores, not p- or q-values.

Value

A `nearender_result`. `scores$rwr` is an FGS-by-AGS matrix whose cells sum steady-state node affinity over the corresponding FGS; `statistics$rwr` is the identical master score matrix. `diagnostics` reports mapped input sizes, convergence, and iteration count; `parameters` records restart, normalization, tolerance, iteration limit, weight use, and identifier case policy.

References

Fang H, Gough J (2014). doi:10.1186/s1307301400648.

See Also

`pagerank.render()`, `nea.render()`, `spectral.render()`

Examples

```
edges <- data.frame(from = c("A", "A", "B"), to = c("B", "C", "C"),
  confidence = c(5, 1, 2))
net <- import_network(edges, "from", "to", "confidence",
  species = NULL, input_id_format = "custom", verbose = FALSE)
ags <- list(sample = "a")
fgs <- list(pathway_b = "b", pathway_c = "c")
rwr <- rwr.render(ags, fgs, net, verbose = FALSE)
rwr$scores$rwr
```

samples2ags *Create altered gene sets from continuous sample profiles*

Description

Convert a gene-by-sample matrix, such as expression, copy-number, protein, or methylation measurements, into one altered gene set (AGS) per sample. `"top"` ranks absolute, unstandardized deviations from each gene's across-sample mean; `"toppos"` ranks signed deviations in descending order, selecting the upper side of the value distribution. Both remain fixed-size methods and apply no significance or positivity cutoff. `"topnorm"` ranks absolute within-gene z-scores. `"bestp"` ranks one-sided upper-tail probabilities calculated from `abs(z)`, whereas `"significant"` retains genes below `cutoff_q` after Benjamini-Hochberg adjustment of the same probabilities. This adjustment is applied to both directions of deviation and does not use the enrichment-only q-value convention of `nea.render()`. `"toprandom"` samples genes without replacement.

Usage

```
samples2ags(
  x,
  n_top = NULL,
  sample_columns = NULL,
  column_pattern = NULL,
  method = c("significant", "top", "toppos", "topnorm", "bestp", "toprandom"),
  cutoff_q = 0.05,
  case_sensitive = FALSE,
  species,
  input_id_format = master_id_format(species),
  id_mappings = NULL,
  ambiguous = c("error", "drop", "first", "expand"),
  unmapped = c("keep", "drop", "error"),
  duplicate_ids = c("error", "mean", "sum", "max")
)
```

Arguments

`x`	Numeric gene-by-sample matrix with gene identifiers as row names.
`n_top`	Number of genes retained by top-ranking methods. Values around 30–300 are often useful; very large sets may reduce specificity.
`sample_columns`	Optional names or indices selecting sample columns.
`column_pattern`	Optional regular expression selecting sample columns.
`method`	Preparation method. `"top"`, `"toppos"`, `"topnorm"`, `"bestp"`, and `"toprandom"` require `n_top`; `"significant"` uses `cutoff_q` and can return an empty AGS.
`cutoff_q`	Benjamini-Hochberg-adjusted probability cutoff for `method = "significant"`.
`case_sensitive`	Use case-sensitive identifier matching. The default treats all gene identifiers case-insensitively.

species — Required species identifier accepted by available_id_formats(). Use explicit NULL for unsupported species or non-biological data.

input_id_format — Submitted identifier format. Defaults to master_id_format() for species; required when species = NULL.

id_mappings — Optional object from load_id_mappings().

ambiguous, unmapped — Mapping policies passed to map_identifiers().

duplicate_ids — Policy for rows that collapse to one identifier after mapping: error, mean, sum, or maximum.

Value

A named list containing one altered gene vector per sample, with normalization metadata available through normalization_info().

See Also

mutations2ags(), import_gene_sets(), save_gs_list()

Examples

```
profiles <- matrix(c(
  9, 1, 2, 8, 5, 5,
  1, 9, 8, 2, 5, 5), nrow = 6,
  dimnames = list(paste0("gene", 1:6), c("sample_1", "sample_2")))
top <- samples2ags(profiles, n_top = 2, method = "top",
  species = NULL, input_id_format = "custom")
toppos <- samples2ags(profiles, n_top = 2, method = "toppos",
  species = NULL, input_id_format = "custom")
top
toppos
```

---

save_gs_list *Save a gene-set list in two-column text format*

Description

Each output row represents one gene or protein and the altered or functional gene set to which it belongs. The tab-delimited file representation can be read by tools that accept NEArender group files, including EviNet. Vector and matrix return modes expose the same representation without writing a file.

Usage

```
save_gs_list(gs_list, file = NULL, output = c("file", "vector", "matrix"))
```

Arguments

gs_list	Named list of gene identifier vectors.
file	Output file path. This must be supplied explicitly when output = "file"; it is ignored for the non-writing return modes.
output	Return mode: "file", "vector", or "matrix". File output is the default, but no default path is assumed.

Value

For file output, the normalized path invisibly; otherwise a named gene vector or a two-column gene-by-group character matrix.

References

Alexeyenko A et al. (2012). doi:10.1186/1471210513226. Jeggari A, Alexeyenko A (2017). doi:10.1186/s128590171534y. Jeggari A et al. (2018). doi:10.1093/nar/gky485.

See Also

import_gene_sets(), samples2ags(), mutations2ags()

Examples

```
sets <- list(first = c("A", "B"), second = c("C", "D"))
save_gs_list(sets, output = "matrix")
group_file <- tempfile(fileext = ".groups")
save_gs_list(sets, file = group_file)
unlink(group_file)
```

---

search_ndex_networks *Search the NDEx network catalog*

Description

Query an NDEx 3 server for network resources and return a stable, tabular catalog. Search is an explicit network operation; it is never called by a renderer or loader. The result can be serialized directly by a web-service adapter, and a one-row result can be passed to ndex_network_summary() or download_ndex_network().

Usage

```
search_ndex_networks(
  query = "*:*",
  account = NULL,
  visibility = c("PUBLIC", "PRIVATE", "all"),
  offset = 0L,
  limit = 100L,
```

```
  server = NULL,
  username = NULL,
  password = NULL,
  token = NULL,
  timeout = 300
)
```

Arguments

`query`	Search text. The default `"*:*"` lists networks visible under the selected filters.
`account`	Optional NDEx account name used to restrict ownership.
`visibility`	Visibility filter: public, private, or all resources the credentials can read. Private searches require authentication.
`offset`	Zero-based result offset.
`limit`	Maximum number of rows, from 1 through 1000. Use pagination for larger catalogs.
`server`	NDEx server root or REST root. Defaults to the public NDEx server. Redirects are not followed. Internet-facing callers must allow-list trusted HTTPS origins rather than passing an arbitrary user-supplied URL.
`username`, `password`	Optional Basic authentication pair.
`token`	Optional bearer token. Do not combine with Basic credentials.
`timeout`	Positive request timeout in seconds.

Value

A `nearender_ndex_catalog` data frame. The `total_hits` attribute reports the number of matches on the server.

See Also

`ndex_network_summary()`, `download_ndex_network()`, `load_ndex_network()`

Examples

```
## Not run:
catalog <- search_ndex_networks("SIGNOR", limit = 20)
catalog[, c("uuid", "name", "owner", "node_count", "edge_count")]

## End(Not run)
```

`set.heat` *Plot a gene-set heat map*

Description

`set.heat()` plots a heat map where the width and height of each cell reflect the sizes of the corresponding gene sets. Cell color represents a supplied score, such as the signed z-score returned by `nea.render()`. If `score = NULL`, color instead represents Jaccard similarity as a measure of gene set member overlap.

Usage

```
set.heat(
  list1 = NULL,
  list2 = NULL,
  score = NULL,
  color_palette = c("grey.red", "blue.white.red"),
  grades = 64L,
  log_score = TRUE,
  log_size = FALSE,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  show_row_dend = TRUE,
  show_col_dend = TRUE,
  hclust_method = "ward.D2",
  title = NULL,
  ...
)
```

Arguments

`list1`	Named list of gene sets represented by heat-map rows, a named numeric vector containing their sizes, or `NULL` to use equal row heights.
`list2`	Named list of gene sets represented by heat-map columns, a named numeric vector containing their sizes, or `NULL` to use equal column widths. Actual gene-set lists are required when `score = NULL`.
`score`	Optional numeric matrix with one row per element of `list1` and one column per element of `list2`. Named dimensions are aligned to the input set names.
`color_palette`	Color scale: `"grey.red"` for a sequential scale or `"blue.white.red"` for a zero-centered diverging scale.
`grades`	Number of colors in the palette.
`log_score`	Apply `sign(x) * log1p(abs(x))` to supplied scores.
`log_size`	Use `log1p()` gene-set sizes to determine cell widths and heights.
`cluster_rows, cluster_cols`	Cluster rows or columns using Euclidean distances between their displayed score profiles.

show_row_dend, show_col_dend
: Show a dendrogram for a clustered dimension.

hclust_method
: Agglomeration method passed to `stats::hclust()`.

title
: Optional plot title.

...
: Reserved for future use.

Details

The function retains the variable-area display of the original NEArender implementation while adding named-matrix alignment, signed score transformation, optional clustering, and an explicit color key.

Value

Invisibly, a list containing the displayed score matrix, original and displayed set sizes, plotting boundaries, clustering objects, and row and column orders.

See Also

`nea.render()`

Examples

```
ags <- list(sample_a = c("A", "B"), sample_b = c("B", "C", "D"))
fgs <- list(pathway_a = c("A", "C"), pathway_b = c("B", "D"))
set.heat(ags, fgs, score = NULL, log_score = FALSE)
scores <- matrix(1:4, 2, dimnames = list(names(ags), names(fgs)))
set.heat(score = scores, log_score = FALSE)
```

---

spectral.render *Render a sample matrix as pathway-level spectral features*

Description

Calculate pathway-level Dirichlet sums and Rayleigh quotients from a sample matrix. Dirichlet sums quantify contrasts between adjacent node states; Rayleigh quotients additionally normalize those contrasts by individual node states.

Usage

```
spectral.render(
  fgs,
  mtr,
  net,
  min_ngenes = 5,
  features = c("rlq", "rlw", "dir", "diw"),
  scope = c("within-pathway", "boundary-expanded"),
```

```
  case_sensitive = FALSE,
  n_cores = 1L,
  verbose = TRUE
)
```

Arguments

fgs	Named list of functional gene sets whose subnetworks are extracted from the global network net.
mtr	Numeric gene-by-sample matrix. Values become sample-specific node attributes.
net	Undirected igraph network, or a legacy network with a named adjacency list in links. Weighted features use the igraph weight edge attribute or legacy named edge-weight vectors in weights.
min_ngenes	Minimum mapped genes per set before scope expansion.
features	Unique spectral features. rlq and dir use binary topology; rlw and diw use stored network edge weights. The default requests all four implemented features.
scope	Graph scope: "within-pathway" uses within-set edges; "boundary-expanded" also uses boundary edges to one-hop neighbors.
case_sensitive	Use case-sensitive identifier matching.
n_cores	Positive integer number of worker processes. Independent spectral features are distributed across workers when n_cores > 1.
verbose	Report progress with message().

Details

The implemented spectral features are Dirichlet sums (dir and diw for edge-unweighted and weighted versions, respectively) and Rayleigh quotients (rlq and rlw for edge-unweighted and weighted versions, respectively). The former summarize contrasts between states of adjacent network nodes within a pathway, whereas the latter also normalize the contrasts by individual node states. A separate pathway-by-sample matrix is produced for each requested feature. Running scope = "boundary-expanded" may take about ten times as long as "within-pathway"; the exact increase depends on the network and pathway sizes. Submit alternative scope analyses as separate calls so that each result has an unambiguous graph scope.

Value

A nearender_result. scores contains one pathway-by-sample matrix for every requested feature, in request order. statistics contains the same master score matrices under the same names. rlq and rlw are Rayleigh quotients; dir and diw are Dirichlet sums. Rows with fewer than min_ngenes mapped genes, or with no usable scoped edges, contain NA. diagnostics$fgs_network_sizes reports mapped pathway sizes; no p- or q-values are calculated.

References

Chung FRK (1997). *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics 92, American Mathematical Society. ISBN 978-0-8218-0315-8.

See Also

spectral_eigenpairs(), texture.render(), gsva.render()

Examples

```
profiles <- matrix(c(1, 3, 5, 2, 2, 2), nrow = 3,
  dimnames = list(c("A", "B", "C"), c("sample_1", "sample_2")))
fgs <- list(pathway = c("A", "B"))
net <- list(
  links = list(A = "B", B = c("A", "C"), C = "B"),
  weights = list(A = c(B = 2), B = c(A = 2, C = 1), C = c(B = 1)),
  Ntotal = 2)
spectral <- spectral.render(fgs, profiles, net, min_ngenes = 2,
  scope = "within-pathway", verbose = FALSE)
spectral$scores$rlq
spectral$scores$rlw
```

---

spectral_eigenpairs *Calculate pathway Laplacian eigenpairs*

Description

Calculate the graph spectrum of each pathway without requiring a function over its nodes. The returned generalized eigenpairs are the structural counterparts of the Rayleigh quotients calculated by spectral.render().

Usage

```
spectral_eigenpairs(
  fgs,
  net,
  min_ngenes = 5,
  use_edge_weights = FALSE,
  scope = c("within-pathway", "boundary-expanded"),
  case_sensitive = FALSE,
  n_cores = 1L,
  verbose = TRUE
)
```

Arguments

fgs	Named list of functional gene sets whose subnetworks are extracted from the global network net.
net	Undirected igraph network, or a legacy network with a named adjacency list in links. When use_edge_weights = TRUE, the igraph weight edge attribute or legacy named edge-weight vectors in weights are required.
min_ngenes	Minimum genes per set that must map to network nodes before scope expansion.

use_edge_weights
: Use the magnitudes of stored network edge weights. The default, FALSE, uses binary topology.

scope
: Graph scope: "within-pathway" uses within-set edges; "boundary-expanded" also includes all one-hop network neighbors of mapped pathway genes.

case_sensitive
: Use case-sensitive identifier matching.

n_cores
: Positive integer number of worker processes. Independent pathways are distributed across workers when n_cores > 1.

verbose
: Report progress with message().

Details

For adjacency matrix A, degree matrix D, and combinatorial Laplacian L = D - A, this function solves the generalized eigenproblem L v = lambda D v. This is equivalent to diagonalizing the symmetric normalized Laplacian D^(-1/2) L D^(-1/2) and transforming its eigenvectors back to the original node coordinates. Returned eigenvectors are therefore D-orthonormal, so their Rayleigh quotients under the corresponding rlq or rlw definition equal their eigenvalues.

Isolated nodes, including nodes incident only to zero-weight edges in a weighted analysis, are removed before decomposition. Negative stored weights are interpreted by magnitude, as in spectral.render(). Because no sample matrix limits node availability, "boundary-expanded" includes every one-hop neighbor represented in net. Eigenvector signs and the basis chosen within a repeated-eigenvalue eigenspace are not unique.

n_cores controls only the distribution of pathways among R worker processes. Each worker calls base::eigen(), and the BLAS/LAPACK library linked to R may independently use all available CPU cores. Actual CPU use can therefore exceed n_cores. Different BLAS/LAPACK implementations and versions can also produce small numerical differences, especially for eigenvectors in repeated or nearly repeated eigenspaces.

Value

A named list in the same order as fgs. Each pathway with at least min_ngenes mapped genes and one usable edge contains an object of class "eigen" with numeric values and vectors. Eigenvalues are in ascending order, rows of vectors are named by retained network nodes, and column i is the eigenvector for values[i]. An ineligible or edge-free pathway has a NULL entry.

References

Chung FRK (1997). *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics 92, American Mathematical Society. ISBN 978-0-8218-0315-8.

See Also

spectral.render()

Examples

```
fgs <- list(pathway = c("A", "B"))
net <- list(
  links = list(A = "B", B = c("A", "C"), C = "B"),
  Ntotal = 2)
pairs <- spectral_eigenpairs(fgs, net, min_ngenes = 2,
  scope = "boundary-expanded", verbose = FALSE)
pairs$pathway$values
rownames(pairs$pathway$vectors)
```

---

string_channel_spec *Describe a released STRING evidence channel*

Description

Describe a released STRING evidence channel

Usage

```
string_channel_spec(
  channel = "textmining",
  version = "12.0",
  network_type = c("functional", "physical")
)
```

Arguments

channel	STRING evidence-channel column.
version	Supported STRING release.
network_type	Functional or physical STRING network.

Value

A resource specification accepted by prepare_resource().

---

texture.render *Render a sample matrix as pathway-level texture features*

Description

From a matrix of continuous gene or protein values, calculate selected Haralick texture features for a collection of pathways. Haralick et al. (1973) developed the method for regular image grids; this implementation evaluates co-occurrence between adjacent nodes in irregular pathway-scale subnetworks extracted from the global network. Subset fgs before calling when only some sets are required.

Usage

```
texture.render(
  fgs,
  mtr,
  net,
  t_level = 3,
  min_ngenes = 5,
  norm_dir = c("row", "col", "non"),
 features = c("asm", "con", "cor", "den", "dva", "ent", "f12", "f13", "idm", "pro",
    "sav", "sen", "sha", "sva", "var"),
  scope = c("within-pathway", "boundary-expanded"),
  case_sensitive = FALSE,
  n_cores = 1L,
  verbose = TRUE
)
```

Arguments

fgs	Named list of functional gene sets whose subnetworks are extracted from the global network. Subset it before calling when needed.
mtr	Numeric gene-by-sample matrix.
net	Network with a named adjacency list in links.
t_level	Number of binary grey-level bits.
min_ngenes	Minimum mapped genes per set.
norm_dir	Rank-normalization direction.
features	Unique requested Haralick feature identifiers. The default explicitly lists all 15 implemented features.
scope	Graph scope: "within-pathway" uses within-set edges; "boundary-expanded" also uses boundary edges to one-hop neighbors.
case_sensitive	Use case-sensitive identifier matching.
n_cores	Positive integer number of worker processes. Independent functional gene sets are distributed across workers when greater than one.
verbose	Report progress with message().

Details

Each sample is discretized to 2^t_level grey levels before a pathway GLCM is constructed. norm_dir controls whether ranking is performed by gene, by sample, or over the full selected matrix. The renderer calculates descriptive texture features; it does not calculate p- or q-values.

Value

A nearender_result. scores contains one pathway-by-sample matrix for every requested Haralick feature, in request order; statistics contains the same master score matrices under the same names. Pathways below min_ngenes retain a row filled with NA. diagnostics$fgs_network_sizes reports the number of mapped network genes for every pathway, while parameters records scope, discretization, normalization direction, case policy, and workers.

Haralick features

`features` accepts the following identifiers:

`asm` angular second moment (energy)

`con` contrast

`cor` correlation

`den` difference entropy

`dva` difference variance

`ent` entropy

`f12` first information measure of correlation

`f13` second information measure of correlation

`idm` inverse difference moment (homogeneity)

`pro` cluster prominence

`sav` sum average

`sen` sum entropy

`sha` cluster shade

`sva` sum variance

`var` variance

References

Haralick RM, Shanmugam K, Dinstein I (1973). doi:10.1109/TSMC.1973.4309314.

Kobayashi K et al. (2015). doi:10.1007/s1008601515076.

Alexeyenko A (2026). doi:10.64898/2026.07.12.737672.

See Also

`glcm.net()`, `haralick.net()`, `haralick.matrix()`, `spectral.render()`, `gsva.render()`

Examples

```
profiles <- matrix(c(1, 5, 2, 4, 3, 3, 4, 2), nrow = 4,
  dimnames = list(letters[1:4], c("sample_1", "sample_2")))
fgs <- list(pathway = c("a", "b", "c"))
net <- list(links = list(a = "b", b = c("a", "c"),
  c = c("b", "d"), d = "c"), Ntotal = 3)
texture <- texture.render(fgs, profiles, net, min_ngenes = 3,
  features = c("asm", "con"), scope = "within-pathway",
  verbose = FALSE)
texture$scores$asm
```

topology_2nd *Investigate second-order network topology*

Description

Visualize the joint distribution of node degrees at opposite endpoints of network edges. Maslov and Sneppen (2002) used degree-preserving randomized networks to report suppressed hub–hub interactions and favored hub–low-degree interactions in the biological networks they studied. `topology_2nd()` displays a two-dimensional kernel density of endpoint degrees; it does not perform that null-model comparison.

Usage

```
topology_2nd(
  nea,
  main = "Second-order topology",
  n = 20L,
  plot = TRUE,
  case_sensitive = FALSE,
  normalize = FALSE
)

topology2nd(...)
```

Arguments

`nea`	Undirected igraph network, or a legacy network with a named adjacency list in `links` and edge count in `Ntotal`.
`main`	Plot title.
`n`	Grid size for the two-dimensional density estimate.
`plot`	Draw the density heat map.
`case_sensitive`	Identifier case policy used when `normalize = TRUE`.
`normalize`	Re-normalize network node identifiers. The default is `FALSE` because networks should be normalized during import. Enabling this option warns because it copies and processes the full adjacency data.
`...`	Arguments passed by the deprecated `topology2nd()` alias.

Value

Invisibly, endpoint data and the density grid.

References

Maslov S, Sneppen K (2002). Specificity and stability in topology of protein networks. Science 296:910–913. doi:10.1126/science.1065103.

See Also

node_degrees(), network_overlap(), nea_member_recovery_benchmark()

Examples

```
net <- list(links = list(
  a = c("b", "c"), b = c("a", "c"), c = c("a", "b", "d"),
  d = c("c", "e"), e = "d"), Ntotal = 5)
topology <- topology_2nd(net, n = 20, plot = FALSE)
head(topology$endpoints)
```

---

universal_member_recovery_benchmark

Benchmark networks and rendering methods by held-out member recovery

---

Description

Randomly split each sufficiently large mapped gene set into training seeds and held-out members. The selected renderer scores every network node against every training set. Held-out members are compared with an equal number of pathway-specific non-member controls.

Usage

```
universal_member_recovery_benchmark(
  net,
  gs,
  method = c("nea", "rwr", "pagerank"),
  renderer_args = list(),
  mask = ".",
  train_fraction = 2/3,
  min_size = 9L,
  match_by_degree = TRUE,
  members = FALSE,
  graph = FALSE,
  seed = NULL,
  case_sensitive = FALSE,
  verbose = TRUE
)
```

Arguments

net, gs, mask, match_by_degree, graph, seed, case_sensitive, verbose	See nea_member_recovery_benchmark().
method	Renderer used for scoring: "nea", "rwr", or "pagerank".
renderer_args	Named list of additional arguments for the renderer. Benchmark-managed inputs and options cannot be overridden.

`train_fraction`	Proportion of mapped set members used as training seeds.
`min_size`	Minimum mapped gene-set size eligible for splitting.
`members`	Whether to return the potentially large training, held-out, and matched-control identifier lists in `splits` and `controls`.

Value

A prediction list accepted by `roc()`, including classification counts and cutoffs as well as the method, renderer arguments, score identifier, train/test splits, matched controls, and split sizes used for reproducibility.

Examples

```
net <- list(links = list(
  a = c("b", "c"), b = c("a", "c"), c = c("a", "b", "d"),
  d = c("c", "e"), e = c("d", "f"), f = c("e", "g"),
  g = c("f", "h"), h = c("g", "i"), i = c("h", "j"),
  j = c("i", "k"), k = "j"), Ntotal = 11)
fgs <- list(pathway = letters[1:9])
prediction <- universal_member_recovery_benchmark(net, fgs,
  method = "nea", min_size = 6, seed = 1, verbose = FALSE)
prediction$split_sizes
```

Index

∗ **Disease Ontology**
    ontology_gene_set_api, 58
∗ **Gene Ontology**
    ontology_gene_set_api, 58
∗ **datasets**
    NEArender.test, 50
    ontology_gene_set_api, 58
∗ **gene sets**
    ontology_gene_set_api, 58
∗ **network enrichment analysis**
    ontology_gene_set_api, 58

as_igraph_network, 4
as_igraph_network(), *5, 6*, *9*, *56*, *69*
as_nearender_network, 5
as_nearender_network(), *5, 6*, *9*
as_sparse_network, 6
as_sparse_network(), *5*, *9*
available_databases, 7
available_databases(), *4*, *10, 11*, *14*, *32*, *34–37*, *41*, *54*, *65*
available_id_formats, 8
available_id_formats(), *4*, *9*, *29*, *31*, *33*, *44, 45*, *47*, *65*, *72*, *81*
available_ontology_collections, 8
available_ontology_collections(), *15*, *42*, *65*

base::eigen(), *88*
benchmark
        *(*nea_member_recovery_benchmark*)*, 52

connectivity *(*node_degrees*)*, 56
convert_network, 9
convert_network(), *5, 6*, *18*, *34*, *72*

disease_ontology_gene_sets
        *(*ontology_gene_set_api*)*, 58
download_database, 10
download_database(), *7*, *11*, *13*, *36*
download_geneset_database, 11
download_geneset_database(), *37*, *65*
download_ndex_network, 12
download_ndex_network(), *40*, *82, 83*
download_network_database, 13
download_network_database(), *42*
download_ontology_collection, 14
download_ontology_collection(), *9*, *43*, *65*
download_resource, 15
download_resource(), *54*, *71*

ear, 16
ear(), *4*, *50*, *66*
export_network, 18
export_network(), *9*, *34*, *72*

filter_ontology_collection, 19
filter_ontology_collection(), *65*

gene_ontology_gene_sets
        *(*ontology_gene_set_api*)*, 58
glcm.net, 20
glcm.net(), *23–25*, *91*
graphics::smoothScatter(), *57*
grepl(), *28*
gsva.render, 21
gsva.render(), *4*, *50*, *66*, *77*, *87*, *91*

haralick.matrix, 22
haralick.matrix(), *25*, *91*
haralick.net, 24
haralick.net(), *21*, *23*, *91*
humanbase_spec, 25
humanbase_spec(), *16*

igraph::page_rank(), *66*
igraph::read_graph(), *72*
igraph::write_graph(), *18*
import.gs *(*import_gene_sets*)*, 27

import.net, 26
import_do_collection, 26
import_do_collection(), *43*, *65*
import_gene_sets, 27
import_gene_sets(), *32*, *36, 37*, *48*, *81, 82*
import_go_collection, 30
import_go_collection(), *43*, *65*
import_kegg_pathways, 31
import_kegg_pathways(), *29*
import_network, 32
import_network(), *9*, *18*, *26*, *36*, *38–42*, *72*

load_database, 35
load_database(), *7*, *10*, *37*, *40*, *43*
load_geneset_database, 37
load_geneset_database(), *11*, *29*, *43*, *65*
load_id_mappings, 38
load_id_mappings(), *29*, *31–33*, *36*, *40, 41*, *44*, *47*, *69–72*, *81*
load_ndex_network, 38
load_ndex_network(), *13*, *83*
load_network_database, 40
load_network_database(), *14*, *34*, *43*, *74*
load_ontology_collection, 42
load_ontology_collection(), *9*, *15*, *43*, *65*

manifest, 43
manifest(), *13*, *36, 37*, *40*, *42*, *46*, *65*, *71*, *73, 74*
map_identifiers, 44
map_identifiers(), *29*, *33*, *47*, *69, 70*, *72*, *81*
master_id_format, 45
master_id_format(), *29*, *33*, *47*, *72*, *81*
merge_networks, 45
merge_networks(), *74*
message(), *17*, *23*, *50*, *53*, *66, 67*, *79*, *86*, *88*, *90*
mutations2ags, 47
mutations2ags(), *4*, *29*, *50*, *81, 82*

ndex_network_summary, 48
ndex_network_summary(), *82, 83*
nea.render, 49
nea.render(), *4*, *17*, *53*, *66, 67*, *74*, *77*, *79, 80*, *84, 85*
nea_member_recovery_benchmark, 52
nea_member_recovery_benchmark(), *54, 55*, *57*, *75, 76*, *93*
NEArender *(*NEArender-package*)*, 3
NEArender-package, 3
NEArender.test, 50
nearender_result, *4*, 51, *76, 77*
network_catalog, 54
network_catalog(), *16*, *70*
network_overlap, 54
network_overlap(), *46*, *57*, *93*
network_qc, 55
node_degrees, 56
node_degrees(), *55*, *93*
normalization_info, 57
normalization_info(), *29*, *31*, *34*, *48*, *81*

ontology_gene_set_api, 58
ora.render, 65
ora.render(), *4*, *17*, *22*, *50*, *77*

pagerank.render, 66
pagerank.render(), *4*, *34*, *50*, *77*, *79*
plot_subgraph, 68
prepare_id_mappings, 69
prepare_id_mappings(), *36*, *41*, *44*, *65*
prepare_resource, 70
prepare_resource(), *15*, *25*, *54*, *74*, *89*

read_network, 71
read_network(), *9*, *18*, *34*
reduce_network, 73
reduce_network(), *46*
render_panel, 74
roc, 74
roc(), *52, 53*, *94*
run_renderer, 76
rwr.render, 78
rwr.render(), *4*, *34*, *50*, *67*, *77*

samples2ags, 80
samples2ags(), *4*, *17*, *29*, *48*, *50*, *82*
save_gs_list, 81
save_gs_list(), *29*, *48*, *81*
search_ndex_networks, 82
search_ndex_networks(), *4*, *12, 13*, *40*, *48*
set.heat, 84
set.heat(), *50*
spectral.render, 85
spectral.render(), *4*, *22*, *50*, *67*, *77*, *79*, *87, 88*, *91*
spectral_eigenpairs, 87
spectral_eigenpairs(), *69*, *87*

stats::hclust(), *85*
string_channel_spec, 89
string_channel_spec(), *16*

texture.render, 89
texture.render(), *4*, *20–25*, *50*, *77*, *87*
topology2nd *(*topology_2nd*)*, 92
topology_2nd, 92
topology_2nd(), *55*, *57*

universal_member_recovery_benchmark, 93

venn::venn(), *55*