Package 'IsoNetAnalyzeR'

November 20, 2026

Title Isoform-Level Protein Interaction Network Analysis

Version 0.99.2

Description Compares predicted protein-interaction neighbourhoods of transcript isoforms. The user supplies isoform seeds labelled ``up'' or ``down'', for example the isoforms that gain or lose usage in a differential transcript usage analysis. Each seed set is expanded in a bundled isoform-resolved predicted interaction network, or in a user-supplied igraph network, and the resulting neighbourhoods are compared per gene and summarised across genes. The package also provides connected-component analysis, network diagnostics, plots, and over-representation and gene set enrichment analysis of neighbourhood statistics. Protein-identical transcripts are resolved to shared representatives, and nonsense-mediated and non-stop decay transcripts, annotated from Ensembl through AnnotationHub, are excluded by default. Results are S4Vectors DataFrames, and gene sets can be GSEABase collections such as those of the msigdb package. Statistical testing of isoform usage remains upstream of the package.

License MIT + file LICENSE

URL https://github.com/IsoformAnalysisGroup/IsoNetAnalyzeR

BugReports https://github.com/IsoformAnalysisGroup/IsoNetAnalyzeR/issues

biocViews Software, Network, GraphAndNetwork, NetworkEnrichment, GeneSetEnrichment, AlternativeSplicing, DifferentialSplicing, Transcriptomics, SystemsBiology, Visualization

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.6.0)

Imports methods, igraph, cli, stats, utils, ggplot2 (>= 3.5.2), ggraph, patchwork, rlang, fgsea, ggrepel, progressr, withr, S4Vectors, AnnotationHub, ensembldb

Suggests BiocStyle, knitr, rmarkdown, msigdb, GSEABase, colorspace, future, future.apply, readxl, SummarizedExperiment, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Collate 'isonetanalyzer_class.R' 'diff_pulldown_gene_class.R' 'diff_pulldown_analysis_class.R' 'graph_utils.R' 'component_analysis.R' 'duplicate_isoforms.R' 'run_component_ora.R' 'run_fgsea.R' 'gsea_accessors.R' 'lookup_class.R' 'lookup_vp.R' 'network_diagnostics_class.R' 'plot_component.R' 'plot_diff_pulldown.R' 'plot_network.R' 'plot_network_diagnostics.R' 'print_metadata.R' 'run_differential_pulldown.R' 'run_network_diagnostics.R' 'transcript_biotypes.R' 'virtual_pulldown.R' 'virtual_pulldown_analysis.R'

Config/roxygen2/version 8.1.1

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

git_branch devel

git_last_commit d0f00e0

git_last_commit_date 2026-11-13

Repository Bioconductor 3.24

Date/Publication 2026-11-19

Author Dimitrios S. Kanakoglou [aut, cre] (ORCID: <https://orcid.org/0000-0001-7735-2029>), Nikolas A. Mumm [aut] (ORCID: <https://orcid.org/0009-0009-5751-0291>), Isabella Østerlund [ctb, ths] (Isoform interaction model; protein-identical isoform map), Kristoffer Vitting-Seerup [cph, ths], Isoform Analysis Group [cph]

Maintainer Dimitrios S. Kanakoglou <dimkan@dtu.dk>

Contents

aggregate_duplicate_counts . . . 3
analyze_subgraph . . . 4
ComponentAnalysis-class . . . 5
ComponentOraResult-class . . . 7
DifferentialPulldownAnalysis-class . . . 8
FgseaResult-class . . . 10
GeneDiffPulldown-class . . . 11
get_analysis . . . 12
get_component . . . 13
get_component_ora_result . . . 14
get_component_overview . . . 15
get_cross_gene_summary . . . 16
get_gene_result . . . 17
get_gsea_result . . . 18
get_gsea_stats . . . 19
get_gsea_table . . . 20
get_isoform_stats . . . 22
get_lookup . . . 23
get_lookup_stats . . . 23
get_lookup_subgraph . . . 24
get_metadata . . . 25

get_network . . . 26
get_network_diagnostics . . . 26
get_ora_table . . . 27
get_pulldown_graph . . . 29
IsoNetAnalyzeR . . . 30
IsoNetAnalyzeR-class . . . 31
isonet_duplicate_map . . . 33
isonet_gene_names . . . 33
isonet_transcript_biotypes . . . 34
list_analyses . . . 35
list_lookups . . . 35
lookup_virtual_pulldown . . . 36
NetworkDiagnostics-class . . . 37
plot_component . . . 39
plot_cross_gene_boxplot . . . 41
plot_cross_gene_boxplot_overview . . . 43
plot_cross_gene_overview . . . 44
plot_cross_gene_scatter . . . 45
plot_network . . . 47
plot_network_diagnostics . . . 51
print_metadata . . . 53
resolve_duplicate_isoforms . . . 53
run_component_analysis . . . 54
run_component_ora . . . 56
run_differential_pulldown . . . 58
run_fgsea . . . 61
run_network_diagnostics . . . 64
set_metadata . . . 65
summarize_subgraphs . . . 66
VirtualPulldownLookup-class . . . 67
virtual_pulldown . . . 68

Index **69**

`aggregate_duplicate_counts` *Aggregate counts for protein-identical isoforms*

Description

Sums raw count rows into the kept transcript representative for each protein-identical group. This transformation must be performed before differential expression or transcript-usage model fitting.

Usage

```
aggregate_duplicate_counts(
  x,
  feature_data = NULL,
  assay = "counts",
  duplicate_map = NULL
)
```

Arguments

x	A numeric matrix, data.frame or DataFrame of raw counts, or a single-assay SummarizedExperiment.
feature_data	Optional data.frame or DataFrame aligned to count rows, containing at least gene_id and isoform_id. When omitted, the rowData of a SummarizedExperiment is used; an ID column it lacks is taken from ENST or gene:ENST row names (genes of mapped transcripts from the duplicate map). A list-valued gene_id, as in tximeta objects, gives each row its first gene. Gene and transcript IDs may carry Ensembl version suffixes.
assay	Assay name used for a SummarizedExperiment; defaults to "counts".
duplicate_map	Optional validated custom map.

Value

An object of class DuplicateCountAggregation: a list with object, feature_data, audit, and summary.

Examples

```
map <- isonet_duplicate_map()
counts <- matrix(
  c(10L, 4L, 7L,
    2L, 1L, 3L,
    5L, 5L, 5L),
  nrow = 3, byrow = TRUE,
  dimnames = list(c(map$kept_ENST[1], map$redundant_ENST[1],
                    map$kept_ENST[2]), paste0("sample", seq_len(3)))
)
features <- data.frame(
  gene_id = c(map$kept_ENSG[1], map$redundant_ENSG[1], map$kept_ENSG[2]),
  isoform_id = rownames(counts)
)
res <- aggregate_duplicate_counts(counts, features)
res$object
res$summary
```

---

analyze_subgraph *Per-node analysis of a virtual pulldown subgraph*

Description

Returns node-level statistics for every vertex in a virtual pulldown subgraph. When multiple seed vertices are present, distance_from_seed is the shortest-path distance to the nearest seed.

Usage

```
analyze_subgraph(graph)
```

Arguments

graph	An igraph object with a pi edge attribute (preserved from the full network after interact == 1 filtering).

Details

Pi statistics (`mean_pi`, `pi_entropy`) are computed only for seed vertices (order == 0), where they characterise the confidence profile of the interactions that anchored the pulldown. For non-seed vertices these columns are `NA` — their edges were pulled in by BFS expansion and the pi values reflect the network globally rather than any seed-specific signal.

Value

A `DataFrame` with one row per vertex and the following columns:

isoform Vertex name (transcript ID).

is_seed Logical flag copied directly from the graph's `order == 0` annotation.

degree Number of edges connected to the vertex.

clustering Local clustering coefficient.

distance_from_seed Shortest-path distance from the nearest seed (0 for a seed). `Inf` indicates the vertex is unreachable from every seed.

mean_pi Mean of the `pi` edge-attribute values across all incident edges. `NA` for non-seed vertices.

pi_entropy Shannon entropy of the normalised `pi` values across all incident edges (bits). Measures how evenly interaction confidence is spread across a seed's partners: low entropy indicates a few high-confidence interactions; high entropy indicates many edges of similar moderate confidence. `NA` for non-seed vertices and for seed vertices with only one incident edge.

Examples

```
obj <- IsoNetAnalyzeR("example")
g <- virtual_pulldown(obj, "ENST00000621271", order = 1)
head(analyze_subgraph(g))
```

ComponentAnalysis-class

ComponentAnalysis - connected-component analysis of a differential pulldown

---

Description

Produced by `run_component_analysis`. Stores Tier-1 structural and Tier-2 topological summary tables derived from decomposing the merged cross-gene network of a DifferentialPulldownAnalysis run.

Usage

```
## S4 method for signature 'ComponentAnalysis'
show(object)
```

Arguments

`object`	A ComponentAnalysis object.

Details

Topology classification (`topology_class`): Each component is heuristically classified based on its degree centralization score:

`"hub-spoke"` Centralization > 0.6: degree is concentrated in one or a few nodes.

`"clique-like"` Centralization < 0.2: connectivity is relatively uniformly distributed.

`"chain-like"` Centralization between 0.2 and 0.6: a intermediate degree concentration.

These thresholds are descriptive heuristics and do not establish a regulator, complex, pathway, or mechanism.

Value

Objects of this class are created by `run_component_analysis`; read them with `get_component_overview` and `get_component`.

Methods (by generic)

- `show(ComponentAnalysis)`: Print a concise summary.

Slots

`label` Character scalar. User-supplied label, used as the `component_analysis` argument in `get_component_overview(analysis, component_analysis)`.

`parent_label` Character scalar. Label of the parent DifferentialPulldownAnalysis run.

`component_table` `DataFrame` (Tier 1). One row per retained connected component. Columns: `component_id`, `n_nodes`, `n_edges`, `density`, `avg_degree`, `clustering`, `n_seeds`, `n_neighbours`, `seed_fraction`, `n_seeds_up`, `n_seeds_down`, `n_genes_up`, `n_genes_down`, `mixed` (logical), `n_shared_nodes`, `shared_fraction`.

`node_table` `DataFrame` (Tier 1). One row per node across all retained components. Columns: `node`, `component_id`, `source_gene`, `is_seed`, `seed_direction` (`"up"`/`"down"`/`"both"`/`NA`), `degree`.

`topology_table` `DataFrame` (Tier 2). One row per retained component. Augments `component_table` with: `component_id`, `diameter`, `radius`, `degree_centralization`, `topology_class` (`"clique-like"`, `"hub-spoke"`, or `"chain-like"` - see Details).

`hub_table` `DataFrame` (Tier 2). One row per shared (`source_gene == "multi-gene"`) node across all retained components, ranked by betweenness centrality within its component. Columns: `node`, `component_id`, `degree`, `betweenness` (raw), `betweenness_norm` (normalised to [0, 1] within each component), `betweenness_rank` (1 = highest).

`direction` Character scalar: `"up"`, `"down"`, or `"both"`.

`level` Character scalar: `"isoform"` or `"gene"`.

`ora_results` Named list of ComponentOraResult objects, keyed by user-supplied label. Populated by `run_component_ora`. Empty list by default.

`genes` Character vector. Gene IDs included (length 0 = all).

`created_at` `POSIXct` scalar. Timestamp of the run.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
obj <- run_component_analysis(obj, "t2d", label = "comp")
get_component_overview(obj, "t2d", "comp")
```

ComponentOraResult-class

ComponentOraResult - GO over-representation analysis on connected components

---

Description

Produced by `run_component_ora`. Stores the ORA result table across all components and all run parameters. Stored inside `ComponentAnalysis@ora_results[[label]]`.

Usage

```
## S4 method for signature 'ComponentOraResult'
show(object)
```

Arguments

`object`	A ComponentOraResult object.

Value

Objects of this class are created by `run_component_ora` and returned by `get_component_ora_result`.

Methods (by generic)

- `show(ComponentOraResult)`: Print a concise summary.

Slots

`label` Character scalar. User-supplied label, used as the `ora_label` argument in `get_component_ora_result(obj, analysis, component_analysis, ora_label)`.

`parent_label` Character scalar. Label of the parent ComponentAnalysis.

`grandparent_label` Character scalar. Label of the parent DifferentialPulldownAnalysis.

`result_table` `DataFrame`. One row per (component, pathway) combination. Columns: `component_id` (integer), `pathway` (character), `pval` (numeric), `padj` (numeric; BH-adjusted within each component), `overlap` (integer; genes in both the component and the pathway), `pathway_size` (integer; pathway size after filtering to universe), `n_genes_tested` (integer; number of genes tested for this component), `overlap_genes` (list column of character vectors; the overlapping gene IDs). Sorted by `component_id` then `padj` ascending.

`universe` Character vector. The background gene set used.

`nodes_used` Character scalar. `"all"` or `"seeds"`.

`go_category` Character scalar. GO subcollection tested, or `NA` when a user-supplied pathway list was used.

`params` Named list. Run parameters: `minSize`, `maxSize`, `n_components_tested`, `n_components_skipped` (too few genes to test), `universe_size`.

`created_at` `POSIXct` scalar. Timestamp of the run.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
obj <- run_component_analysis(obj, "t2d", label = "comp")
# Two small gene sets of gene symbols, for illustration. MSigDB collections
# are available as GeneSetCollections from the msigdb package; see the
# vignette.
gene_sets <- GSEABase::GeneSetCollection(list(
  GSEABase::GeneSet(c("WAS", "VAV1", "ICOS"), setName = "set_A",
                    geneIdType = GSEABase::SymbolIdentifier()),
  GSEABase::GeneSet(c("CLTCL1", "PRKACB", "HLA-DMA"), setName = "set_B",
                    geneIdType = GSEABase::SymbolIdentifier())
))
obj <- run_component_ora(obj, "t2d", "comp", label = "example",
                         pathways = gene_sets, minSize = 2)
get_component_ora_result(obj, "t2d", "comp", "example")
```

DifferentialPulldownAnalysis-class

DifferentialPulldownAnalysis - full differential pulldown analysis result

---

Description

Top-level container produced by `run_differential_pulldown`. Stores one GeneDiffPulldown object per gene, the validated input table, the run parameters, and cross-gene summary statistics.

Usage

```
## S4 method for signature 'DifferentialPulldownAnalysis'
show(object)
```

Arguments

`object` A DifferentialPulldownAnalysis object.

Value

Objects of this class are created by `run_differential_pulldown` and returned by `get_analysis`.

Methods (by generic)

- `show(DifferentialPulldownAnalysis)`: Print a concise run summary.

Slots

`label` Character scalar. A user-supplied label for this analysis run, used as the `label` argument in `get_analysis(obj, label)`.

`input_table` `DataFrame`. The validated long-format input table with columns `gene_id`, `isoform_id`, `direction` (`"up"` / `"down"`). Protein-identical collisions have been collapsed to resolved representatives.

`gene_results` Named list of GeneDiffPulldown objects, one per gene in `input_table`. Names are gene IDs.

`cross_gene_summary` `DataFrame` with one row per gene summarising up/down subgraph properties side by side. Columns: `gene_id`, `n_up_input`, `n_down_input` (cleaned distinct submitted isoforms), `n_up_resolved`, `n_down_resolved` (resolved protein representatives analyzed), `n_up_found`, `n_down_found` (resolved representatives present in the network before BFS expansion), `pct_up_found`, `pct_down_found` (percentage of resolved representatives found in the network; `NA` when `n_resolved == 0`), `up_nodes`, `up_edges`, `up_density`, `up_avg_degree`, `up_clustering`, `up_starness` (degree centralization of the up subgraph; 0 = uniform, 1 = perfect hub-and-spoke topology), `up_genes` (number of unique genes represented in the up subgraph), `down_nodes`, `down_edges`, `down_density`, `down_avg_degree`, `down_clustering`, `down_starness` (degree centralization of the down subgraph), `down_genes` (number of unique genes represented in the down subgraph), `n_shared_nodes` (nodes present in both up and down subgraphs), `n_up_unique` (nodes exclusive to the up subgraph), `n_down_unique` (nodes exclusive to the down subgraph), `jaccard` (Jaccard index of node-set overlap between up and down subgraphs: `|up intersect down| / |up union down|`; `NA` when either subgraph is absent), `n_edges_gained` (edges unique to the up-seed neighbourhood), `n_edges_lost` (edges unique to the down-seed neighbourhood), `n_edges_shared` (edges present in both subgraphs; together with `n_edges_gained` and `n_edges_lost` this partitions the edges in the two seed-derived neighbourhoods). All three edge columns are `NA` when either subgraph is absent, `up_seed_mean_pi` (mean of per-seed mean-pi values across all up seeds; summarises average interaction confidence of the up seed set), `up_seed_pi_entropy` (mean of per-seed pi-entropy values across all up seeds; summarises how evenly interaction confidence is distributed across partners for the up seed set), `down_seed_mean_pi`, `down_seed_pi_entropy` (analogous metrics for the down seed set). All four columns are `NA` when the corresponding subgraph is absent or carries no `pi` edge attribute. Node, edge and gene counts, their overlaps and `jaccard` include the seeds, so a direction with more seeds counts more even when both reach the same partners.

`gsea_results` Named list of FgseaResult objects, keyed by user-supplied label. Populated by `run_fgsea`. Empty list by default.

`component_analyses` Named list of ComponentAnalysis objects, keyed by user-supplied label. Populated by `run_component_analysis`. Empty list by default.

`order` Integer scalar. Number of BFS hops used.

`cutoff` Numeric scalar. Internal-edge fraction cutoff used.

`created_at` `POSIXct` scalar. Timestamp of the run.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name == "CLTCL1", ],
  order = 1, confirm = FALSE, verbose = FALSE
)
get_analysis(obj, "t2d")
```

FgseaResult-class *FgseaResult - GSEA result on a differential pulldown analysis*

Description

Produced by run_fgsea. Stores the ranked gene vector, the fgsea result table, and all run parameters. One FgseaResult object can be stored per user-supplied label inside a DifferentialPulldownAnalysis; retrieve with `get_gsea_result(obj, analysis, label)`.

Usage

```
## S4 method for signature 'FgseaResult'
show(object)
```

Arguments

`object`	A FgseaResult object.

Value

Objects of this class are created by `run_fgsea` and returned by `get_gsea_result`.

Methods (by generic)

- `show(FgseaResult)`: Print a concise summary.

Slots

`label` Character scalar. User-supplied label, used as the `gsea_label` argument in `get_gsea_result(obj, analysis, gsea_label)`.

`parent_label` Character scalar. Label of the parent DifferentialPulldownAnalysis run.

`result_table` DataFrame. Output of `fgsea::fgsea()` with standard columns: `pathway`, `pval`, `padj`, `ES`, `NES`, `size`, `leadingEdge`, and `log2err` (`fgseaMultilevel`) or `nMoreExtreme` (`fgseaSimple`). Sorted by `padj` ascending. Access via `get_gsea_table(obj, analysis, gsea_label)`.

`ranked_genes` Named numeric vector. The gene-level rank statistic passed to fgsea, sorted descending. Names are gene IDs matching the pathway gene sets. Access as a table via `get_gsea_stats(obj, analysis, gsea_label)`.

`rank_by` Character scalar. The preset used (`rank_by` argument), or `"custom"` when the user supplied `stats` directly.

`go_category` Character scalar. GO subcollection(s) and MSigDB release fetched with the msigdb package, or `NA` when gene sets were supplied.

`params` Named list. Run parameters: `minSize`, `maxSize`, `nperm` (`NULL` unless set), `method` (`"fgseaMultilevel"` or `"fgseaSimple"`), `n_genes_ranked`, `n_genes_dropped_na`.

`created_at` `POSIXct` scalar. Timestamp of the run.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
# Six genes for speed; the vignette analyses all 65. With few ranked genes,
# minSize must be small.
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
# Two small gene sets of gene symbols, for illustration. MSigDB collections
# are available as GeneSetCollections from the msigdb package; see the
# vignette.
gene_sets <- GSEABase::GeneSetCollection(list(
  GSEABase::GeneSet(c("WAS", "VAV1", "ICOS"), setName = "set_A",
                    geneIdType = GSEABase::SymbolIdentifier()),
  GSEABase::GeneSet(c("CLTCL1", "PRKACB", "HLA-DMA"), setName = "set_B",
                    geneIdType = GSEABase::SymbolIdentifier())
))
set.seed(42)
obj <- run_fgsea(obj, "t2d", label = "example_gsea", pathways = gene_sets,
                 minSize = 3)
get_gsea_result(obj, "t2d", "example_gsea")
```

GeneDiffPulldown-class

GeneDiffPulldown - per-gene differential pulldown result

Description

Stores the virtual pulldown subgraphs and statistics for a single gene's up- and down-labelled isoform seeds. One GeneDiffPulldown object is created per gene in a DifferentialPulldownAnalysis run.

Usage

```
## S4 method for signature 'GeneDiffPulldown'
show(object)
```

Arguments

`object` A GeneDiffPulldown object.

Value

Objects of this class are returned by `get_gene_result`.

Methods (by generic)

- `show(GeneDiffPulldown)`: Print a concise per-gene summary.

Slots

`gene_id` Character scalar. The gene identifier from the input table.

`up_isoforms` Character vector. Isoform IDs from the input table that were labelled up for this gene.

`down_isoforms` Character vector. Isoform IDs from the input table that were labelled down for this gene.

`up_graph` An `igraph` object. Virtual pulldown subgraph seeded from `up_isoforms`, or `NULL` when no up-labelled isoforms were present in the network.

`down_graph` An `igraph` object. Virtual pulldown subgraph seeded from `down_isoforms`, or `NULL` when no down-labelled isoforms were present in the network.

`up_stats` `DataFrame`. Per-node statistics for `up_graph` (output of `analyze_subgraph`), or a zero-row `DataFrame` when `up_graph` is `NULL`.

`down_stats` `DataFrame`. Per-node statistics for `down_graph`, or a zero-row `DataFrame` when `down_graph` is `NULL`.

`graph_summary` `DataFrame` with one row per direction (`"up"`, `"down"`) summarising network-level properties: nodes, edges, density, avg_degree, clustering, starness, genes. Rows are `NA` when the corresponding graph is absent.

`gene_summary` Named list of gene-level overview scalars: `n_up_input`, `n_down_input` (cleaned distinct submitted isoforms), `n_up_resolved`, `n_down_resolved` (resolved protein representatives analyzed), `n_up_network`, `n_down_network` (isoforms found in the network), `has_up`, `has_down` (logical flags for subgraph availability).

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name == "CLTCL1", ],
  order = 1, confirm = FALSE, verbose = FALSE
)
get_gene_result(obj, "t2d", "ENSG00000070371")
```

`get_analysis` *Retrieve a stored analysis result*

Description

Retrieve a stored analysis result

Usage

```
get_analysis(obj, label)
```

Arguments

obj	An IsoNetAnalyzeR session object.
label	Character scalar. The key under which the analysis was stored.

Value

A DifferentialPulldownAnalysis object.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name == "CLTCL1", ],
  order = 1, confirm = FALSE, verbose = FALSE
)
list_analyses(obj)
get_analysis(obj, "t2d")
```

---

get_component *Get a merged per-node table for a single connected component*

Description

Returns a single DataFrame with one row per node in the requested component, combining the Tier-1 node membership columns (@node_table) with the Tier-2 hub betweenness columns (@hub_table) via a left-join on node and component_id.

Usage

```
get_component(obj, analysis, component_analysis, component_id)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the parent DifferentialPulldownAnalysis; use list_analyses(obj) to see available labels.
component_analysis	Character scalar. Label of the ComponentAnalysis; use list_analyses(obj) then inspect the result to see available component analysis labels.
component_id	Integer scalar. The component to retrieve, as listed in component_id column of get_component_overview. Component IDs are assigned in descending order of node count (edge count as tiebreaker), so component_id = 1 is always the largest component.

Details

Nodes that are not shared across genes (`source_gene != "multi-gene"`) receive `NA` in the hub columns (`betweenness`, `betweenness_norm`, `betweenness_rank`), as betweenness is only computed for shared nodes. This makes it easy to filter for hub nodes with `!is.na(betweenness_norm)` or to sort all nodes by their hub score.

Value

A `DataFrame` with one row per node in the requested component and the following columns: `node`, `component_id`, `source_gene`, `is_seed`, `seed_direction`, `degree`, `betweenness` (`NA` for non-shared nodes), `betweenness_norm` (`NA` for non-shared nodes), `betweenness_rank` (`NA` for non-shared nodes). Rows are sorted by `betweenness_norm` descending (hub nodes first), with `NA`s at the end.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
obj <- run_component_analysis(obj, "t2d", label = "comp")

# Nodes of the largest component
comp1 <- get_component(obj, "t2d", "comp", component_id = 1)
head(comp1)

# Seed nodes only
comp1[comp1$is_seed, ]
```

---

get_component_ora_result

Retrieve a stored component ORA result

Description

Retrieve a stored component ORA result

Usage

```
get_component_ora_result(obj, analysis, component_analysis, ora_label)

## S4 method for signature 'IsoNetAnalyzeR'
get_component_ora_result(obj, analysis, component_analysis, ora_label)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`analysis`	Character scalar. Label of the parent DifferentialPulldownAnalysis.
`component_analysis`	Character scalar. Label of the ComponentAnalysis.
`ora_label`	Character scalar. Label of the ComponentOraResult.

Value

A ComponentOraResult object.

Methods (by class)

- `get_component_ora_result(IsoNetAnalyzeR)`: Method for IsoNetAnalyzeR.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
obj <- run_component_analysis(obj, "t2d", label = "comp")
# Two small gene sets of gene symbols, for illustration. MSigDB collections
# are available as GeneSetCollections from the msigdb package; see the
# vignette.
gene_sets <- GSEABase::GeneSetCollection(list(
  GSEABase::GeneSet(c("WAS", "VAV1", "ICOS"), setName = "set_A",
                    geneIdType = GSEABase::SymbolIdentifier()),
  GSEABase::GeneSet(c("CLTCL1", "PRKACB", "HLA-DMA"), setName = "set_B",
                    geneIdType = GSEABase::SymbolIdentifier())
))
obj <- run_component_ora(obj, "t2d", "comp", label = "example",
                         pathways = gene_sets, minSize = 2)
ora <- get_component_ora_result(obj, "t2d", "comp", "example")
ora
```

`get_component_overview` *Get a merged per-component overview table from a ComponentAnalysis result*

Description

Returns a single wide `DataFrame` with one row per retained connected component, combining all Tier-1 structural columns (`@component_table`) and all Tier-2 topological columns (`@topology_table`) side by side via a left-join on `component_id`.

Usage

```
get_component_overview(obj, analysis, component_analysis)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the parent DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
component_analysis	Character scalar. Label of the ComponentAnalysis; use `list_analyses(obj)` then inspect the result to see available component analysis labels.

Details

This is the recommended starting point for exploring a ComponentAnalysis result: scan the table for mixed components (`mixed == TRUE`), hub-spoke topology (`topology_class == "hub-spoke"`), or high shared-node fractions (`shared_fraction`), then drill into individual components with `get_component`.

Value

A `DataFrame` with one row per retained component and the following columns (Tier-1 first, then Tier-2): `component_id`, `n_nodes`, `n_edges`, `density`, `avg_degree`, `clustering`, `n_seeds`, `n_neighbours`, `seed_fraction`, `n_seeds_up`, `n_seeds_down`, `n_genes_up`, `n_genes_down`, `mixed`, `n_shared_nodes`, `shared_fraction`, `diameter`, `radius`, `degree_centralization`, `topology_class`. Topology columns are `NA` for any component absent from `@topology_table` (e.g. single-node components where topology metrics are undefined).

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
obj <- run_component_analysis(obj, "t2d", label = "comp")
overview <- get_component_overview(obj, "t2d", "comp")

# Largest components first
head(overview[order(-overview$n_nodes),
              c("component_id", "n_nodes", "n_edges", "topology_class")])
```

---

get_cross_gene_summary

Get the cross-gene summary table from a differential pulldown analysis

---

Description

Returns the wide-format summary `DataFrame` with one row per gene, containing subgraph topology metrics for both the up- and down-labelled conditions side by side.

Usage

```
get_cross_gene_summary(obj, analysis)

## S4 method for signature 'IsoNetAnalyzeR'
get_cross_gene_summary(obj, analysis)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`analysis`	Character scalar. Label of the DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.

Value

A `DataFrame` with one row per gene. See DifferentialPulldownAnalysis for a full description of columns.

Methods (by class)

- `get_cross_gene_summary(IsoNetAnalyzeR)`: Method for IsoNetAnalyzeR.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
cgs <- get_cross_gene_summary(obj, "t2d")
cgs[, c("gene_id", "up_nodes", "down_nodes", "up_edges", "down_edges")]
```

`get_gene_result` *Get the per-gene result object from a differential pulldown analysis*

Description

Returns the GeneDiffPulldown object for a single gene, which contains the up and down subgraphs, per-node statistics, and graph-level summaries for that gene.

Usage

```
get_gene_result(obj, analysis, gene_id)

## S4 method for signature 'IsoNetAnalyzeR'
get_gene_result(obj, analysis, gene_id)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
gene_id	Character scalar. Ensembl gene ID (or whichever identifier was used in the input table).

Value

A GeneDiffPulldown object.

Methods (by class)

- `get_gene_result(IsoNetAnalyzeR)`: Method for IsoNetAnalyzeR.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name == "CLTCL1", ],
  order = 1, confirm = FALSE, verbose = FALSE
)
gr <- get_gene_result(obj, "t2d", "ENSG00000070371")
gr
```

---

get_gsea_result *Retrieve a stored GSEA result from a differential pulldown analysis*

Description

Retrieve a stored GSEA result from a differential pulldown analysis

Usage

```
get_gsea_result(obj, analysis, gsea_label)

## S4 method for signature 'IsoNetAnalyzeR'
get_gsea_result(obj, analysis, gsea_label)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the DifferentialPulldownAnalysis.
gsea_label	Character scalar. Label of the FgseaResult stored inside the analysis.

Value

A FgseaResult object.

Methods (by class)

- `get_gsea_result(IsoNetAnalyzeR)`: Method for IsoNetAnalyzeR.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
# Six genes for speed; the vignette analyses all 65. With few ranked genes,
# minSize must be small.
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
# Two small gene sets of gene symbols, for illustration. MSigDB collections
# are available as GeneSetCollections from the msigdb package; see the
# vignette.
gene_sets <- GSEABase::GeneSetCollection(list(
  GSEABase::GeneSet(c("WAS", "VAV1", "ICOS"), setName = "set_A",
                    geneIdType = GSEABase::SymbolIdentifier()),
  GSEABase::GeneSet(c("CLTCL1", "PRKACB", "HLA-DMA"), setName = "set_B",
                    geneIdType = GSEABase::SymbolIdentifier())
))
set.seed(42)
obj <- run_fgsea(obj, "t2d", label = "example_gsea", pathways = gene_sets,
                 minSize = 3)
get_gsea_result(obj, "t2d", "example_gsea")
```

---

get_gsea_stats *Get the ranked gene statistic from a GSEA result as a table*

Description

Returns the named numeric vector of rank statistics that was passed to `fgsea::fgsea()` as a `DataFrame`, making it easy to inspect which genes drove the enrichment, join with other tables, or plot a ranking barcode.

Usage

```
get_gsea_stats(obj, analysis, gsea_label)

## S4 method for signature 'IsoNetAnalyzeR'
get_gsea_stats(obj, analysis, gsea_label)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`analysis`	Character scalar. Label of the DifferentialPulldownAnalysis.
`gsea_label`	Character scalar. Label of the FgseaResult.

Value

A DataFrame with columns:

gene_id Gene identifier (names from the ranked vector).

stat Rank statistic value (e.g. delta_nodes).

rank Integer rank (1 = highest statistic).

Sorted by stat descending (i.e. rank 1 first).

Methods (by class)

- get_gsea_stats(IsoNetAnalyzeR): Method for IsoNetAnalyzeR.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
# Six genes for speed; the vignette analyses all 65. With few ranked genes,
# minSize must be small.
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
# Two small gene sets of gene symbols, for illustration. MSigDB collections
# are available as GeneSetCollections from the msigdb package; see the
# vignette.
gene_sets <- GSEABase::GeneSetCollection(list(
  GSEABase::GeneSet(c("WAS", "VAV1", "ICOS"), setName = "set_A",
                    geneIdType = GSEABase::SymbolIdentifier()),
  GSEABase::GeneSet(c("CLTCL1", "PRKACB", "HLA-DMA"), setName = "set_B",
                    geneIdType = GSEABase::SymbolIdentifier())
))
set.seed(42)
obj <- run_fgsea(obj, "t2d", label = "example_gsea", pathways = gene_sets,
                 minSize = 3)

# The ranked statistic used by fgsea, highest first
head(get_gsea_stats(obj, "t2d", "example_gsea"))
```

---

get_gsea_table *Get the GSEA result table from a differential pulldown analysis*

Description

Returns the fgsea::fgsea() result table stored in a FgseaResult object as a DataFrame, with an optional significance filter.

Usage

```
get_gsea_table(obj, analysis, gsea_label, padj_cutoff = NULL)

## S4 method for signature 'IsoNetAnalyzeR'
get_gsea_table(obj, analysis, gsea_label, padj_cutoff = NULL)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the DifferentialPulldownAnalysis.
gsea_label	Character scalar. Label of the FgseaResult stored inside the analysis.
padj_cutoff	Numeric scalar or NULL. When supplied, only rows with padj <= padj_cutoff are returned. Default NULL (return all pathways).

Value

A DataFrame with columns pathway, pval, padj, ES, NES, size, leadingEdge, and log2err (fgseaMultilevel) or nMoreExtreme (fgseaSimple), sorted by padj ascending.

Methods (by class)

- get_gsea_table(IsoNetAnalyzeR): Method for IsoNetAnalyzeR.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
# Six genes for speed; the vignette analyses all 65. With few ranked genes,
# minSize must be small.
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
# Two small gene sets of gene symbols, for illustration. MSigDB collections
# are available as GeneSetCollections from the msigdb package; see the
# vignette.
gene_sets <- GSEABase::GeneSetCollection(list(
  GSEABase::GeneSet(c("WAS", "VAV1", "ICOS"), setName = "set_A",
                    geneIdType = GSEABase::SymbolIdentifier()),
  GSEABase::GeneSet(c("CLTCL1", "PRKACB", "HLA-DMA"), setName = "set_B",
                    geneIdType = GSEABase::SymbolIdentifier())
))
set.seed(42)
obj <- run_fgsea(obj, "t2d", label = "example_gsea", pathways = gene_sets,
                 minSize = 3)

# All tested gene sets, and those passing an adjusted p-value cutoff
get_gsea_table(obj, "t2d", "example_gsea")
get_gsea_table(obj, "t2d", "example_gsea", padj_cutoff = 0.05)
```

`get_isoform_stats` *Get combined isoform-level statistics for a gene from a differential pulldown analysis*

Description

Merges the per-node statistics from the up- and down-labelled subgraphs of a single gene into one long-format `DataFrame`, with a `direction` column (`"up"` / `"down"`) identifying the condition. A `is_seed` column flags isoforms that were supplied as seeds, using the explicit seed annotation stored by `analyze_subgraph()`; for results saved without it, the gene's stored up and down seed isoforms. Isoforms that appear in only one condition are represented in that condition's rows only; no artificial NAs are introduced for the missing direction.

Usage

```
get_isoform_stats(obj, analysis, gene_id)

## S4 method for signature 'IsoNetAnalyzeR'
get_isoform_stats(obj, analysis, gene_id)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`analysis`	Character scalar. Label of the DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
`gene_id`	Character scalar. Ensembl gene ID (or whichever identifier was used in the input table).

Value

A `DataFrame` with one row per isoform per direction and columns: `direction`, `isoform`, `is_seed`, `degree`, `clustering`, `distance_from_seed`, `mean_pi`, `pi_entropy`.

Methods (by class)

- `get_isoform_stats(IsoNetAnalyzeR)`: Method for IsoNetAnalyzeR.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name == "CLTCL1", ],
  order = 1, confirm = FALSE, verbose = FALSE
)
stats <- get_isoform_stats(obj, "t2d", "ENSG00000070371")
head(stats)
```

get_lookup *Retrieve a stored lookup result*

Description

Retrieve a stored lookup result

Usage

```
get_lookup(obj, label)
```

Arguments

obj	An IsoNetAnalyzeR session object.
label	Character scalar. The key under which the lookup was stored.

Value

A VirtualPulldownLookup object.

Examples

```
obj <- IsoNetAnalyzeR("example")
obj <- lookup_virtual_pulldown(obj, "cltcl1_up", seed = "ENST00000621271",
                               order = 1)
get_lookup(obj, "cltcl1_up")
```

get_lookup_stats *Get the per-node statistics from a virtual pulldown look-up*

Description

Returns the `node_stats` `DataFrame` stored in a VirtualPulldownLookup object directly, without having to unpack the S4 slot manually. It contains one row per node in the pulldown subgraph and is the output of `analyze_subgraph`.

Usage

```
get_lookup_stats(obj, label)

## S4 method for signature 'IsoNetAnalyzeR'
get_lookup_stats(obj, label)
```

Arguments

obj	An IsoNetAnalyzeR session object.
label	Character scalar. Label of the VirtualPulldownLookup; use `list_lookups(obj)` to see available labels.

Value

A DataFrame with one row per node in the subgraph and columns produced by analyze_subgraph: isoform, is_seed, degree, clustering, distance_from_seed, mean_pi, pi_entropy.

Methods (by class)

- get_lookup_stats(IsoNetAnalyzeR): Method for IsoNetAnalyzeR.

Examples

```
obj <- IsoNetAnalyzeR("example")
obj <- lookup_virtual_pulldown(obj, "cltcl1_up", seed = "ENST00000621271",
                               order = 1)
stats <- get_lookup_stats(obj, "cltcl1_up")
head(stats)
```

---

get_lookup_subgraph *Get the subgraph from a virtual pulldown look-up*

Description

Returns the igraph subgraph stored in a VirtualPulldownLookup object directly, without having to unpack the S4 slot manually. The subgraph can be passed directly to plot_network(), igraph functions, or any other graph-level utility.

Usage

```
get_lookup_subgraph(obj, label)

## S4 method for signature 'IsoNetAnalyzeR'
get_lookup_subgraph(obj, label)
```

Arguments

obj	An IsoNetAnalyzeR session object.
label	Character scalar. Label of the VirtualPulldownLookup; use list_lookups(obj) to see available labels.

Value

An igraph object representing the pulldown subnetwork.

Methods (by class)

- get_lookup_subgraph(IsoNetAnalyzeR): Method for IsoNetAnalyzeR.

Examples

```
obj <- IsoNetAnalyzeR("example")
obj <- lookup_virtual_pulldown(obj, "cltcl1_up", seed = "ENST00000621271",
                               order = 1)
g <- get_lookup_subgraph(obj, "cltcl1_up")
c(nodes = igraph::vcount(g), edges = igraph::ecount(g))
```

`get_metadata` *Retrieve metadata from an IsoNetAnalyzeR session object*

Description

Returns one named entry of the session metadata, or the complete metadata list. The constructor and the analysis functions record provenance here: `network_source`, `network_cutoff` and `network_biotype_filter` (the transcript eligibility audit) are set when the session is created, and `analysis_network_coverage`, `analysis_seed_support` and `analysis_duplicate_resolution` are lists keyed by analysis label.

Usage

```
get_metadata(obj, key = NULL)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`key`	Character scalar naming one metadata entry, or `NULL` (default) to return all entries.

Value

The value stored under `key`, or the named list of all metadata when `key` is `NULL`.

See Also

`set_metadata`, `print_metadata`

Examples

```
obj <- IsoNetAnalyzeR("example")
names(get_metadata(obj))
get_metadata(obj, "network_cutoff")
get_metadata(obj, "network_biotype_filter")$excluded_nmd
```

get_network *Retrieve the interaction network of an IsoNetAnalyzeR session*

Description

Returns the isoform interaction network used by the session, after transcript eligibility filtering.

Usage

```
get_network(obj)
```

Arguments

obj	An IsoNetAnalyzeR session object.

Value

An undirected igraph object. Vertices are transcripts with name, gene and transcript_biotype attributes; edges carry a binary interact attribute and, for the bundled network, the predicted interaction probability pi.

Examples

```
obj <- IsoNetAnalyzeR("example")
net <- get_network(obj)
c(vertices = igraph::vcount(net), edges = igraph::ecount(net))
```

get_network_diagnostics *Retrieve the network diagnostics result from an IsoNetAnalyzeR object*

Description

Returns the NetworkDiagnostics object stored in the session, or stops with an informative message if run_network_diagnostics has not been called yet.

Usage

```
get_network_diagnostics(obj)

## S4 method for signature 'IsoNetAnalyzeR'
get_network_diagnostics(obj)
```

Arguments

obj	An IsoNetAnalyzeR session object.

Value

A NetworkDiagnostics object.

Methods (by class)

- `get_network_diagnostics(IsoNetAnalyzeR)`: Method for IsoNetAnalyzeR.

Examples

```
# A small deterministic network keeps this example fast; on the bundled
# network the diagnostics take about half a minute.
g <- igraph::make_graph("Zachary")
n <- igraph::vcount(g)
igraph::V(g)$name <- paste0("tx", seq_len(n))
igraph::V(g)$gene <- paste0("gene", ceiling(seq_len(n) / 2))
igraph::V(g)$transcript_biotype <- "protein_coding"
igraph::E(g)$interact <- 1L
toy <- IsoNetAnalyzeR("toy", network = g,
                      metadata = list(network_cutoff = 0.25))
toy <- run_network_diagnostics(toy)
get_network_diagnostics(toy)
```

---

get_ora_table *Get the ORA result table from a component analysis*

Description

Returns the `fgsea::fora()` result table stored in a ComponentOraResult object as a `DataFrame`. Results can be filtered by component, significance threshold, and/or restricted to the top N terms per component. All filters compose — they are applied in the order: `component_id` → `padj_cutoff` → `top_n`.

Usage

```
get_ora_table(
  obj,
  analysis,
  component_analysis,
  ora_label,
  component_id = NULL,
  padj_cutoff = NULL,
  top_n = NULL
)

## S4 method for signature 'IsoNetAnalyzeR'
get_ora_table(
  obj,
  analysis,
  component_analysis,
  ora_label,
  component_id = NULL,
  padj_cutoff = NULL,
  top_n = NULL
)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the parent DifferentialPulldownAnalysis.
component_analysis	Character scalar. Label of the ComponentAnalysis.
ora_label	Character scalar. Label of the ComponentOraResult.
component_id	Integer scalar or NULL. When supplied, only rows for that component are returned. Default NULL (all components).
padj_cutoff	Numeric scalar or NULL. When supplied, only rows with padj <= padj_cutoff are returned. Default NULL (no filter).
top_n	Integer scalar or NULL. When supplied, only the top N terms per component (sorted by padj ascending, ties broken by pval) are returned. Applied after padj_cutoff. Default NULL (all terms).

Value

A DataFrame with columns component_id, pathway, pval, padj, overlap, pathway_size, n_genes_tested, overlap_genes. Sorted by component_id then padj ascending.

Methods (by class)

- get_ora_table(IsoNetAnalyzeR): Method for IsoNetAnalyzeR.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
obj <- run_component_analysis(obj, "t2d", label = "comp")
# Two small gene sets of gene symbols, for illustration. MSigDB collections
# are available as GeneSetCollections from the msigdb package; see the
# vignette.
gene_sets <- GSEABase::GeneSetCollection(list(
  GSEABase::GeneSet(c("WAS", "VAV1", "ICOS"), setName = "set_A",
                    geneIdType = GSEABase::SymbolIdentifier()),
  GSEABase::GeneSet(c("CLTCL1", "PRKACB", "HLA-DMA"), setName = "set_B",
                    geneIdType = GSEABase::SymbolIdentifier())
))
obj <- run_component_ora(obj, "t2d", "comp", label = "example",
                         pathways = gene_sets, minSize = 2)

# Top two gene sets per component
get_ora_table(obj, "t2d", "comp", "example", top_n = 2)

# Significant gene sets of component 1
get_ora_table(obj, "t2d", "comp", "example", component_id = 1,
              padj_cutoff = 0.05)
```

`get_pulldown_graph` *Retrieve the up- or down-seeded pulldown graph of one gene*

Description

Returns the virtual pulldown subgraph seeded from a gene's up-labelled or down-labelled isoforms.

Usage

```
get_pulldown_graph(x, direction = c("up", "down"))

## S4 method for signature 'GeneDiffPulldown'
get_pulldown_graph(x, direction = c("up", "down"))
```

Arguments

`x`	A GeneDiffPulldown object, as returned by `get_gene_result`.
`direction`	Either `"up"` or `"down"`.

Value

An `igraph` object, or `NULL` when none of the gene's isoforms in that direction were found in the session network. A seed that is in the network but has no predicted interaction gives a one-vertex graph.

Methods (by class)

- `get_pulldown_graph(GeneDiffPulldown)`: Method for GeneDiffPulldown.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name == "CLTCL1", ],
  order = 1, confirm = FALSE, verbose = FALSE
)
cltcl1 <- get_gene_result(obj, "t2d", "ENSG00000070371")
up <- get_pulldown_graph(cltcl1, "up")
c(nodes = igraph::vcount(up), edges = igraph::ecount(up))
```

`IsoNetAnalyzeR` *Create a new IsoNetAnalyzeR session object*

Description

Initialises an IsoNetAnalyzeR object. When no `network` is supplied the package's internal isoform interaction network (`.network_data`) is loaded automatically, along with the associated `.network_cutoff` and `.pi_threshold` values stored as metadata.

Usage

```
IsoNetAnalyzeR(
  name = "IsoNetAnalyzeR",
  network = NULL,
  metadata = list(),
  include_nmd = FALSE,
  include_nsd = FALSE,
  transcript_biotypes = NULL
)
```

Arguments

`name`	Character scalar. A human-readable label for the session (e.g. `"my_experiment"`). Defaults to `"IsoNetAnalyzeR"`.
`network`	An `igraph` object to use as the interaction network. Pass `NULL` (the default) to load the package's built-in network. Repeated edges between the same two transcripts are merged into one (`interact` is 1 when any copy is 1; numeric attributes such as `pi` are averaged over the predicted copies, or over all copies when none is predicted) and self-loops are removed, because they would inflate the degrees used by the neighbourhood cutoff.
`metadata`	Named list of additional metadata to attach to the session. This is merged with any automatically populated fields (`pi_threshold`, `network_cutoff`) - user-supplied values take precedence on name collision.
`include_nmd`	Include nonsense-mediated decay transcripts. Default FALSE.
`include_nsd`	Include non-stop decay transcripts. Default FALSE. Exclusions apply to seeds and neighbours. Set both flags TRUE for an unfiltered network.
`transcript_biotypes`	Optional transcript annotation: a `data.frame` or `DataFrame` with `isoform_id` and `transcript_biotype` columns, which overrides vertex annotations, or an `EnsDb` object. A network with Ensembl transcript IDs (`ENST`) is always annotated from Ensembl release 107 (the GENCODE v41 gene build of the bundled network), loaded once per R session from AnnotationHub record AH104864 and cached on disk by AnnotationHub, or from the `EnsDb` given here; a `transcript_biotype` vertex attribute and a table override it for their transcripts. The full annotation is needed so that seeds outside the network that are decay transcripts are excluded too. If it cannot be loaded (for example offline), a session is still created when both `include_nmd` and `include_nsd` are `TRUE` (without biotypes and gene symbols), or when every vertex has a biotype from the attribute or table (with a warning that decay transcripts outside the network cannot be blocked).

Networks without Ensembl transcript IDs do not load the annotation when the session is created; gene-symbol labels of Ensembl gene IDs in plots and printed summaries can still load it. Unknown vertex biotypes are an error when either decay exclusion is enabled. Gene-level biotypes are not used.

Value

An IsoNetAnalyzeR object. The exclusion audit is stored in metadata$network_biotype_filter. Create a new session to change the policy. When biotypes come from an EnsDb, the network also gets a gene_name vertex attribute with gene symbols, used for plot labels.

Examples

```
# Session on the bundled network (NMD and NSD transcripts excluded)
obj <- IsoNetAnalyzeR("example")
obj

# A user-supplied network needs vertex names, a `gene` vertex attribute,
# a binary `interact` edge attribute and known transcript biotypes
g <- igraph::graph_from_data_frame(
  data.frame(from = "tx1", to = "tx2", interact = 1L, pi = 0.9),
  directed = FALSE,
  vertices = data.frame(name = c("tx1", "tx2"), gene = c("g1", "g2"),
                        transcript_biotype = "protein_coding")
)
IsoNetAnalyzeR("custom", network = g, metadata = list(network_cutoff = 0))
```

IsoNetAnalyzeR-class *IsoNetAnalyzeR - main session object*

Description

The central object of the **IsoNetAnalyzeR** package. It holds the isoform interaction network, session metadata, a named list of full analyses (e.g. virtual pulldown runs), and a named list of quick look-ups.

Usage

```
## S4 method for signature 'IsoNetAnalyzeR'
show(object)

## S4 method for signature 'IsoNetAnalyzeR'
list_analyses(obj)

## S4 method for signature 'IsoNetAnalyzeR'
get_analysis(obj, label)

## S4 method for signature 'IsoNetAnalyzeR'
list_lookups(obj)

## S4 method for signature 'IsoNetAnalyzeR'
get_lookup(obj, label)
```

Arguments

object	An IsoNetAnalyzeR object.
obj	An IsoNetAnalyzeR session object.
label	Character scalar. The key under which the lookup was stored.

Value

Objects of this class are created by IsoNetAnalyzeR.

Methods (by generic)

- show(IsoNetAnalyzeR): Print a concise summary of an IsoNetAnalyzeR session object.
- list_analyses(IsoNetAnalyzeR): List labels of stored analyses.
- get_analysis(IsoNetAnalyzeR): Retrieve a stored analysis by label.
- list_lookups(IsoNetAnalyzeR): List labels of stored lookups.
- get_lookup(IsoNetAnalyzeR): Retrieve a stored lookup by label.

Slots

name Character scalar. A user-defined label for the session.

network An igraph object representing the isoform interaction network. Populated from the package's internal data by default; can be replaced with a user-supplied graph in the future.

created_at POSIXct scalar. Timestamp recorded when the object was constructed.

metadata Named list of arbitrary session-level metadata (e.g. pi_threshold, network_cutoff carried over from internal data, or user-supplied fields).

analyses Named list of DifferentialPulldownAnalysis objects (and potentially other analysis classes added in the future).

lookups Named list of VirtualPulldownLookup objects (and potentially other look-up classes added in the future).

network_diagnostics Either NULL (before diagnostics have been run) or a NetworkDiagnostics object produced by run_network_diagnostics.

Examples

```
obj <- IsoNetAnalyzeR("example")
obj
list_analyses(obj)
```

isonet_duplicate_map *Protein-identical isoform identifier map*

Description

Returns the transcriptome-wide annotation used by IsoNetAnalyzeR to map redundant Ensembl transcript identifiers to the representative transcript retained for the same protein product. Rows outside the embedded network are retained because they are still needed for upstream count aggregation.

Usage

```
isonet_duplicate_map(refresh = FALSE)
```

Arguments

refresh	Logical; reread the packaged CSV instead of using the package-local cache.

Value

A DataFrame with kept/redundant ENSG, ENSP, and ENST identifiers and a network_applicable flag. Build provenance is in S4Vectors::metadata(map)$provenance.

Examples

```
map <- isonet_duplicate_map()
head(map[, c("kept_ENST", "redundant_ENST", "network_applicable")])
table(map$network_applicable)
```

isonet_gene_names *Gene symbols for Ensembl gene IDs*

Description

Returns the gene name, usually the HGNC symbol, of every gene in an Ensembl annotation; by default Ensembl release 107 from AnnotationHub (record AH104864), as for isonet_transcript_biotypes. Plots use it to label nodes, and it can annotate any result table keyed by Ensembl gene ID.

Usage

```
isonet_gene_names(edb = NULL)
```

Arguments

edb	An EnsDb object, or NULL (default) for the Ensembl 107 annotation from AnnotationHub.

Value

A DataFrame with gene_id (Ensembl, without version) and gene_name columns. gene_name is NA for genes without a symbol.

Examples

```
gene_names <- isonet_gene_names()
gene_names[gene_names$gene_id == "ENSG00000070371", ]
```

---

isonet_transcript_biotypes

Transcript biotypes used for protein-network eligibility

Description

Returns the transcript biotype of every transcript in an Ensembl annotation. By default this is Ensembl release 107, the gene build of GENCODE v41 used by the bundled network, loaded from AnnotationHub (record AH104864) and cached for the R session. These transcript biotypes are not gene-level coding labels or evidence of stable protein production.

Usage

```
isonet_transcript_biotypes(edb = NULL)
```

Arguments

edb	An EnsDb object, or NULL (default) for the Ensembl 107 annotation from AnnotationHub.

Value

A DataFrame with isoform_id and transcript_biotype columns.

Examples

```
biotypes <- isonet_transcript_biotypes()
head(biotypes)
head(sort(table(biotypes$transcript_biotype), decreasing = TRUE))
```

list_analyses *List stored analysis labels*

Description

List stored analysis labels

Usage

```
list_analyses(obj)
```

Arguments

obj	An IsoNetAnalyzeR session object.

Value

Character vector of analysis labels, or `character(0)` if none.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name == "CLTCL1", ],
  order = 1, confirm = FALSE, verbose = FALSE
)
list_analyses(obj)
```

list_lookups *List stored lookup labels*

Description

List stored lookup labels

Usage

```
list_lookups(obj)
```

Arguments

obj	An IsoNetAnalyzeR session object.

Value

Character vector of lookup labels, or `character(0)` if none.

Examples

```
obj <- IsoNetAnalyzeR("example")
obj <- lookup_virtual_pulldown(obj, "cltcl1_up", seed = "ENST00000621271",
                               order = 1)
list_lookups(obj)
```

`lookup_virtual_pulldown` *Perform a virtual pulldown look-up with one or more seeds*

Description

Runs a virtual pulldown for one or more seed isoforms or genes and adds the result as a VirtualPulldownLookup object to the IsoNetAnalyzeR object. All supplied seeds are treated as order-0 anchors in a single pulldown. The network is taken directly from `obj`; the cutoff defaults to the value stored in `obj@metadata[["network_cutoff"]]` but can be overridden to explore alternative thresholds.

Usage

```
lookup_virtual_pulldown(
  obj,
  label,
  seed,
  order,
  cutoff = NULL,
  level = c("isoform", "gene"),
  resolve_duplicates = TRUE
)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`label`	Character scalar. Key under which to store the look-up.
`seed`	Character vector. One or more isoform vertex names or gene names. When multiple seeds are given they are all used as order-0 anchors in the same virtual pulldown.
`order`	Integer. Number of interaction hops.
`cutoff`	Numeric in [0, 1]. Internal-edge fraction threshold. Falls back to `obj@metadata[["network_cutoff` when `NULL`. Supply an explicit value to explore alternative thresholds without modifying the session object.
`level`	Character scalar. `"isoform"` (default) or `"gene"`.
`resolve_duplicates`	Logical. For isoform-level lookups, resolve versioned and protein-identical redundant transcript IDs against the active network. Default `TRUE`.

Value

An updated IsoNetAnalyzeR object (invisibly); access the look-up via `get_lookup(obj, label)`.

Examples

```
obj <- IsoNetAnalyzeR("example")

# One isoform seed, using the session's stored cutoff
obj <- lookup_virtual_pulldown(obj, "cltcl1_up", seed = "ENST00000621271",
                               order = 1)

# Several seeds act as joint order-0 anchors; try a stricter cutoff
obj <- lookup_virtual_pulldown(
  obj, "cltcl1_both",
  seed = c("ENST00000621271", "ENST00000427926"),
  order = 1, cutoff = 0.4
)

# All network isoforms of a gene as seeds
obj <- lookup_virtual_pulldown(obj, "cltcl1_gene", seed = "ENSG00000070371",
                               order = 1, level = "gene")
list_lookups(obj)
get_lookup(obj, "cltcl1_gene")
```

NetworkDiagnostics-class

NetworkDiagnostics - network characterisation result

Description

Top-level container produced by `run_network_diagnostics`. Stores structural and topological summary statistics of the loaded isoform interaction network. Stored in the `@network_diagnostics` slot of an IsoNetAnalyzeR session object; retrieved with `get_network_diagnostics`.

Usage

```
## S4 method for signature 'NetworkDiagnostics'
show(object)
```

Arguments

`object`	A NetworkDiagnostics object.

Value

Objects of this class are created by `run_network_diagnostics` and returned by `get_network_diagnostics`.

Methods (by generic)

- `show(NetworkDiagnostics)`: Print a concise diagnostics summary.

Slots

`network_stats` `DataFrame` with one row summarising the three network views and global topology metrics. Columns:

- **full_isoforms** Total isoform nodes in the full network.
- **full_genes** Unique genes in the full network.
- **full_edges** Total edges in the session network. For the bundled network (from 0.99.0) this equals the number of predicted interaction edges.
- **full_predicted_edges** Edges with `interact == 1`.
- **pred_isoforms** Isoform nodes in the predicted subnetwork.
- **pred_genes** Unique genes in the predicted subnetwork.
- **pred_edges** Edges in the predicted subnetwork.
- **pred_density** Edge density of the predicted isoform subnetwork.
- **pred_avg_degree** Mean isoform degree in the predicted subnetwork.
- **pred_avg_clustering** Mean local clustering coefficient.
- **pred_n_components** Number of connected components.
- **pred_largest_component_n** Node count of the largest component.
- **pred_largest_component_pct** Percentage of nodes in the largest component.
- **pred_diameter** Diameter of the largest component.
- **pred_avg_path_length** Mean shortest path length in the largest component.
- **gene_nodes** Nodes in the gene-level predicted network.
- **gene_edges** Edges in the gene-level predicted network.
- **isolated_isoforms** Isoforms with zero predicted edges.
- **isolated_isoforms_pct** Percentage of all isoforms that are isolated.

`isoform_degree_table` `DataFrame` with one row per isoform in the predicted subnetwork. Columns: `isoform_id`, `gene_id`, `degree`. Isoforms with zero predicted edges are included with `degree = 0`.

`gene_degree_table` `DataFrame` with one row per gene in the gene-level predicted network. Columns: `gene_id`, `degree`.

`component_table` `DataFrame` with one row per connected component in the predicted isoform subnetwork, sorted descending by size. Columns: `component_id`, `n_nodes`, `n_edges`.

`clustering_table` `DataFrame` with one row per isoform in the predicted subnetwork. Columns: `isoform_id`, `gene_id`, `degree`, `clustering_coefficient`.

`isoform_per_gene_table` `DataFrame` with one row per gene present in the full network. Columns: `gene_id`, `n_isoforms_total` (all isoforms for this gene in the full network), `n_isoforms_connected` (isoforms with at least one predicted edge), `n_isoforms_isolated` (isoforms with zero predicted edges), `connectivity_status` (character: `"all_connected"`, `"partial"`, or `"all_isolated"`).

`sticky_isoform_table` `DataFrame` with one row per sticky isoform (degree above threshold). Columns: `isoform_id`, `gene_id`, `degree`. Empty if no sticky isoforms are found.

`sticky_gene_table` `DataFrame` with one row per sticky gene. Columns: `gene_id`, `degree`. Empty if none are found.

`sticky_method` Character scalar. Threshold method used: `"quantile"` or `"sd"`.

`sticky_cutoff` Numeric scalar. Cutoff value supplied to the threshold method.

`sticky_isoform_threshold` Numeric scalar. The computed degree threshold above which an isoform is classified as sticky.

`sticky_gene_threshold` Numeric scalar. The computed degree threshold above which a gene is classified as sticky.

`created_at` `POSIXct` scalar. Timestamp of the run.

Examples

```
# A small deterministic network keeps this example fast; on the bundled
# network the diagnostics take about half a minute.
g <- igraph::make_graph("Zachary")
n <- igraph::vcount(g)
igraph::V(g)$name <- paste0("tx", seq_len(n))
igraph::V(g)$gene <- paste0("gene", ceiling(seq_len(n) / 2))
igraph::V(g)$transcript_biotype <- "protein_coding"
igraph::E(g)$interact <- 1L
toy <- IsoNetAnalyzeR("toy", network = g,
                      metadata = list(network_cutoff = 0.25))
toy <- run_network_diagnostics(toy)
get_network_diagnostics(toy)
```

---

plot_component *Plot a single connected component from a ComponentAnalysis result*

Description

Extracts one connected component by its `component_id` (as assigned in `@component_table`) and renders it using the same cross-gene plot style as `plot_network` on a DifferentialPulldownAnalysis object.

Usage

```
plot_component(
  obj,
  analysis,
  component_analysis,
  component_id,
  colour_by = c("gene", "hub_degree", "direction"),
  colour_neighbours = FALSE,
  layout = "stress",
  node_size = 3,
  label_size = 2.5,
  edge_alpha = 0.4,
  title = NULL,
  node_labels = c("symbol", "id", "none"),
  ...
)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`analysis`	Character scalar. Label of the parent DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
`component_analysis`	Character scalar. Label of the ComponentAnalysis; use `list_analyses(obj)` then inspect the result to see available component analysis labels.

component_id — Integer scalar. The component to plot, matching the component_id column in get_component_overview().

colour_by — Character scalar: "gene" (default), "hub_degree", or "direction".

colour_neighbours — Logical. Only used when colour_by = "direction" and level = "isoform". When TRUE neighbour nodes are coloured by nearest-seed territory instead of neutral grey. Default FALSE.

layout — ggraph layout string. Default "stress".

node_size — Numeric. Base node size. Default 3.

label_size — Numeric. Label text size. Default 2.5.

edge_alpha — Numeric in [0, 1]. Edge opacity. Default 0.4.

title — Character scalar or NULL (auto-generated).

node_labels — Character scalar. "symbol" (default) labels nodes with gene symbols, "id" with transcript (or, at gene level, gene) identifiers, and "none" draws no labels. See plot_network.

... — Additional arguments forwarded to ggraph::ggraph().

Details

Component IDs are assigned in descending order of node count (edge count as tiebreaker), so component_id = 1 is always the largest component. These IDs match the component_id column in all four ComponentAnalysis tables.

Value

A ggplot object, invisibly.

Colour modes (colour_by)

"gene" Each node is coloured by the pulldown gene it belongs to. Shared (multi-gene) nodes are dark grey.

"hub_degree" Shared nodes are coloured by degree on a sequential (Mako) scale; all other nodes are light grey.

"direction" Encodes the regulation direction of each node. Behaviour differs by analysis level — see below.

Direction mode — isoform level

Seed nodes are coloured by their caller-supplied direction: up (green) and down (orange), the up and down colours of every plot, or present in both directions (purple). Neighbour behaviour is controlled by colour_neighbours:

FALSE **(default)** Neighbours are light grey.

TRUE **— territory map** Each neighbour is coloured by the direction of its nearest seed group, computed as unweighted shortest-path distance within the component. Neighbours equidistant from both seed groups are shown in purple (equidistant territory).

Direction mode — gene level

At gene level each node represents a collapsed gene rather than a single isoform, so a binary up/down label is not meaningful (most genes contribute both up- and down-labelled isoforms). Instead a continuous **dominance score** is computed per gene:

$$score = \frac{n_{up} - n_{down}}{n_{up} + n_{down}} \in [-1, +1]$$

where n_{up} and n_{down} are the numbers of seed isoforms from the input table that are up- and down-labelled for that gene. The score is mapped to a diverging green–grey–orange colour scale:

- **+1** (pure up) → green
- **0** (balanced or neighbour) → light grey
- **-1** (pure down) → orange

Neighbour genes (no seed isoforms in this component) receive a score of 0 and appear light grey. `colour_neighbours` is ignored at gene level.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
obj <- run_component_analysis(obj, "t2d", label = "comp")

# Colour seeds by direction
p <- plot_component(obj, "t2d", "comp", component_id = 1,
                    colour_by = "direction")

# Territory map: neighbours coloured by the seed directions reaching them
p <- plot_component(obj, "t2d", "comp", component_id = 1,
                    colour_by = "direction", colour_neighbours = TRUE)

# Without node labels
p <- plot_component(obj, "t2d", "comp", component_id = 1,
                    node_labels = "none")
```

---

plot_cross_gene_boxplot

Plot cross-gene summary: boxplot of the difference between two columns

---

Description

For each gene computes a per-gene delta (`x_col - y_col`, or the log10 fold-change `log10(x_col / y_col)` when `log_ratio = TRUE`) and displays the distribution as a boxplot with individual gene points overlaid as a jitter strip. A horizontal reference line at zero marks no difference between the two columns. Green points have a positive delta (x_col > y_col), orange a negative one and grey none.

Usage

```
plot_cross_gene_boxplot(
  obj,
  analysis,
  x_col,
  y_col,
  log_ratio = FALSE,
  label_n = 10L,
  label = "delta",
  title = NULL
)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
x_col	Character scalar. Column for the minuend / numerator (e.g. the up-seed neighbourhood metric). Must be numeric.
y_col	Character scalar. Column for the subtrahend / denominator (e.g. the down-seed neighbourhood metric). Must be numeric. Delta = `x_col - y_col` (or `log10(x_col/y_col)` when `log_ratio = TRUE`).
log_ratio	Logical. When `TRUE`, plot `log10((x_col + 1) / (y_col + 1))` instead of `x_col - y_col`. Zeros map to 0 and no genes are dropped. Default `FALSE`.
label_n	Integer. Maximum total genes to label across all strategies. Default `10`. Set `0` to suppress all labels.
label	Character vector. One or more of `"delta"` (largest absolute delta), `"up"` (largest positive delta, green points), `"down"` (largest negative delta, orange points). With several strategies, their top genes are labelled in turn. Default `"delta"`.
title	Character scalar or `NULL` (auto-generated).

Value

A ggplot object, invisibly.

Log-ratio mode

When `log_ratio = TRUE` the plotted value is `log10((x_col + 1) / (y_col + 1))`. Adding 1 before dividing means zero-value genes map to 0 (no change) rather than being dropped, so all genes are retained. The zero reference line corresponds to a ratio of 1 (no difference).

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
```

```
p <- plot_cross_gene_boxplot(obj, "t2d", "up_nodes", "down_nodes")
```

`plot_cross_gene_boxplot_overview` *Overview boxplot: five delta distributions in a single row*

Description

Assembles five `plot_cross_gene_boxplot` panels in one row, each showing the distribution of per-gene deltas (up - down) for a standard metric pair:

1. Nodes
2. Edges
3. Density
4. Average degree
5. Clustering coefficient

All panels include `expand_limits(y = 0)` so the zero reference line is always in frame, making it easy to compare the direction of shift across metrics. Labels are suppressed by default for a clean overview; use `plot_cross_gene_boxplot` to drill into any panel with labels.

Usage

```
plot_cross_gene_boxplot_overview(
  obj,
  analysis,
  log_ratio = FALSE,
  label_n = 0L,
  label = "delta",
  title = NULL
)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`analysis`	Character scalar. Label of the DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
`log_ratio`	Logical. Use `log10((x+1)/(y+1))` instead of `x - y`? Default `FALSE`. When `TRUE`, zeros map to 0 (no change) and no genes are dropped.
`label_n`	Integer. Maximum total genes to label per panel. Default `0`.
`label`	Character vector. One or more of `"delta"` (largest absolute delta), `"up"` (largest positive delta, green points), `"down"` (largest negative delta, orange points). With several strategies, their top genes are labelled in turn. Default `"delta"`.
`title`	Character scalar or `NULL` (auto-generated).

Details

When `log_ratio = TRUE`, the delta is computed as `log10((x+1)/(y+1))` so that zero-value genes map to 0 rather than being dropped.

Requires the **patchwork** package.

Value

A patchwork object, invisibly.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
p <- plot_cross_gene_boxplot_overview(obj, "t2d")
```

---

plot_cross_gene_overview

Overview scatter: six standard up-vs-down comparisons in a 2x3 grid

---

Description

A first-look function that assembles six plot_cross_gene_scatter panels in a 2x3 grid:

1. Nodes (up vs down)
2. Edges (up vs down)
3. Density (up vs down)
4. Average degree (up vs down)
5. Clustering coefficient (up vs down)
6. Edges gained vs edges lost

Count columns (nodes, edges, edges gained/lost) use log10(x + 1) scaling by default so that zero-interaction genes stay at the origin rather than being dropped. Labels are suppressed so the overview stays clean; use plot_cross_gene_scatter directly on any panel that catches your eye to add labels and explore further.

Usage

```
plot_cross_gene_overview(
  obj,
  analysis,
  log_counts = TRUE,
  label_n = 0L,
  label = "delta",
  title = NULL
)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
log_counts	Logical. Apply `log10(x + 1)` scaling to the nodes, edges, and edges-gained/lost panels? Default `TRUE`. Zeros map to 0 and are retained on the plot.
label_n	Integer. Maximum total genes to label per panel. Default `0` (no labels - keeps the overview clean). Set to a small value like `3` to flag the most extreme genes.
label	Character vector. Labelling strategy passed to each panel. One or more of `"delta"`, `"x"`, `"y"`. Only used when `label_n > 0`. Default `"delta"`.
title	Character scalar or `NULL`. Overall title for the combined figure. Defaults to `"Cross-gene overview [<analysis>]"`.

Details

Requires the **patchwork** package (`install.packages("patchwork")`).

Value

A `patchwork` object, invisibly. The combined figure is printed as a side-effect.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
p <- plot_cross_gene_overview(obj, "t2d")
```

---

plot_cross_gene_scatter

Plot cross-gene summary: scatter of two columns per gene

Description

Plots any two numeric columns from `cross_gene_summary` as a scatter, one point per gene. A diagonal identity line makes up/down asymmetry immediately visible. For a quick five-panel overview across all standard metrics, see `plot_cross_gene_overview`.

Usage

```
plot_cross_gene_scatter(
  obj,
  analysis,
  x_col,
  y_col,
  x_axis_log = FALSE,
  y_axis_log = FALSE,
  x_log_offset = 1,
  y_log_offset = 1,
  label_n = 10L,
  label = "delta",
  title = NULL
)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
x_col	Character scalar. Column for the X-axis (must be numeric in `cross_gene_summary`).
y_col	Character scalar. Column for the Y-axis (must be numeric in `cross_gene_summary`).
x_axis_log	Logical. Apply `log10(x + 1)` scaling to the X-axis? Default `FALSE`. Zeros map to 0 and are retained; the axis label will read `log10(<col> + 1)`.
y_axis_log	Logical. Apply `log10(x + 1)` scaling to the Y-axis? Default `FALSE`.
x_log_offset	Numeric scalar. Offset added before `log10()` on the x-axis when `x_axis_log = TRUE`. Default 1, giving `log10(x + 1)`.
y_log_offset	Numeric scalar. Offset added before `log10()` on the y-axis when `y_axis_log = TRUE`. Default 1.
label_n	Integer. Maximum total genes to label across all strategies. Default `10`. Set `0` to suppress all labels.
label	Character vector. One or more of `"delta"` (genes furthest from the identity diagonal), `"x"` (highest X values), `"y"` (highest Y values). With several strategies, their top genes are labelled in turn, so `c("delta", "x")` labels both. Default `"delta"`.
title	Character scalar or `NULL` (auto-generated).

Value

A `ggplot` object, invisibly.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
```

```
)
p <- plot_cross_gene_scatter(obj, "t2d", "up_nodes", "down_nodes")
```

`plot_network` *Plot an isoform interaction network*

Description

Renders a network using **ggraph** and **ggplot2**. Dispatches on IsoNetAnalyzeR (full session network) and VirtualPulldownLookup (single pulldown subgraph).

Dispatches on GeneDiffPulldown. Three display modes are available via the `direction` argument:

- `"up"` or `"down"` - renders a single subgraph, highlighting the seed isoforms.
- `"side_by_side"` (default) - renders the two subgraphs side-by-side as a two-panel **ggplot2** / **patchwork** figure.
- `"overlay"` - merges the up- and down-labelled subgraphs into a single network. Non-seed nodes and edges use green for the up-seed neighbourhood, orange for the down-seed neighbourhood, and blue when present in both. Focal seeds use a reddish-purple fill and directional seed shapes, separating node role from neighbourhood membership. Down-only edges are dashed and shared edges thicker, and the legend sits below the plot.

Dispatches on DifferentialPulldownAnalysis. Merges the virtual pulldown subgraphs from multiple genes into a single network to reveal cross-gene clustering and shared interaction partners.

Usage

```
plot_network(
  object,
  level = "isoform",
  all_edges = NULL,
  group_by_gene = FALSE,
  layout = "stress",
  node_size = 3,
  label_size = 2.5,
  edge_alpha = 0.4,
  title = NULL,
  node_labels = c("symbol", "id", "none"),
  ...
)

## S4 method for signature 'IsoNetAnalyzeR'
plot_network(
  object,
  level = "isoform",
  all_edges = NULL,
  group_by_gene = FALSE,
  layout = "stress",
  node_size = 3,
  label_size = 2.5,
  edge_alpha = 0.4,
```

```
  title = NULL,
  node_labels = c("symbol", "id", "none"),
  force = FALSE,
  ...
)

## S4 method for signature 'VirtualPulldownLookup'
plot_network(
  object,
  level = "isoform",
  all_edges = NULL,
  group_by_gene = FALSE,
  layout = "stress",
  node_size = 3,
  label_size = 2.5,
  edge_alpha = 0.4,
  title = NULL,
  node_labels = c("symbol", "id", "none"),
  ...
)

## S4 method for signature 'GeneDiffPulldown'
plot_network(
  object,
  level = "isoform",
  all_edges = NULL,
  group_by_gene = FALSE,
  layout = "stress",
  node_size = 3,
  label_size = 2.5,
  edge_alpha = 0.4,
  title = NULL,
  node_labels = c("symbol", "id", "none"),
  ...,
  direction = c("side_by_side", "up", "down", "overlay")
)

## S4 method for signature 'DifferentialPulldownAnalysis'
plot_network(
  object,
  level = "isoform",
  all_edges = NULL,
  group_by_gene = FALSE,
  layout = "stress",
  node_size = 3,
  label_size = 2.5,
  edge_alpha = 0.4,
  title = NULL,
  node_labels = c("symbol", "id", "none"),
  ...,
  direction = c("both", "up", "down"),
  genes = NULL,
```

```
  min_nodes = NULL,
  min_shared_nodes = NULL,
  min_component_nodes = NULL,
  min_edges = NULL,
  colour_by = c("gene", "hub_degree")
)
```

Arguments

object	A DifferentialPulldownAnalysis object.
level	"isoform" (default) or "gene".
all_edges	Deprecated and ignored; a value other than the default NULL gives a warning.
group_by_gene	Logical. Only used when level = "isoform". Colours nodes by their parent gene when TRUE. Default FALSE.
layout	ggraph layout string. Default "stress".
node_size	Numeric. Base node size. Default 3.
label_size	Numeric. Label text size. Default 2.5.
edge_alpha	Numeric in [0, 1]. Default 0.4.
title	Character scalar or NULL (auto-generated).
node_labels	Character scalar. "symbol" (default) labels nodes with gene symbols, "id" with transcript (or, at gene level, gene) identifiers, and "none" draws no labels. Symbols come from a gene_name vertex attribute when the network has one, otherwise from isonet_gene_names(); nodes without a known symbol keep their identifier.
...	Additional arguments forwarded to ggraph::ggraph().
force	Logical. If TRUE, bypass the size guard and attempt to plot the full network regardless of its size. Default FALSE.
direction	Character scalar: "up" (merge all up-labelled subgraphs), "down" (merge all down-labelled subgraphs), or "both" (default; union of up and down per gene). Must be passed as a named argument.
genes	Character vector of gene IDs to include. Defaults to NULL, which includes all genes in the analysis. Must be passed as a named argument.
min_nodes	Integer or NULL (default). Pre-merge per-gene filter: genes whose merged subgraph (union of up + down vertices for the requested direction) contains fewer than min_nodes nodes are dropped before the cross-gene graph is built. Useful for removing genes with very small or absent pulldown neighborhoods. Set to NULL to disable. Must be passed as a named argument.
min_shared_nodes	Integer or NULL (default). Post-merge component filter: connected components of the merged graph that contain fewer than min_shared_nodes multi-gene nodes (nodes attributed to more than one gene's pulldown) are removed. This is the strongest lever for keeping only biologically interesting cross-gene hubs. Set to NULL to disable. Must be passed as a named argument.
min_component_nodes	Integer or NULL (default). Post-merge component filter: connected components with fewer than min_component_nodes total nodes are removed after merging. Complements min_shared_nodes by also dropping small single-gene islands. Set to NULL to disable. Must be passed as a named argument.

`min_edges` Integer or `NULL` (default). Post-merge component filter: connected components with fewer than `min_edges` edges are removed. Helps discard loosely connected star-shaped or chain components that pass a node-count threshold. Set to `NULL` to disable. Must be passed as a named argument.

`colour_by` Character scalar: `"gene"` (default) or `"hub_degree"`. Controls the node colour encoding.

`"gene"` Each node is coloured by the gene whose pulldown it originated from (Okabe-Ito for N <= 8, HCL Dynamic for larger N). Multi-gene shared nodes are shown in dark grey; their size is scaled by degree so high-connectivity hubs stand out.

`"hub_degree"` Non-shared nodes are collapsed to light grey. Shared multi-gene nodes are coloured by degree on a sequential Mako scale (dark = low degree, cream = high degree), making cross-gene interaction hubs the visual focus of the plot.

Must be passed as a named argument.

Details

Gene-level plot: Isoforms are contracted by the `gene` vertex attribute. Isoforms with a missing `gene` attribute are grouped as `"NA_gene"`. Isolated seed nodes are retained and rendered as valid one-node/zero-edge results; isolated non-seed nodes are omitted.

Node shapes and black outlines provide redundant role encoding alongside colour.

Nodes are coloured by the gene whose pulldown they originated from. Nodes that appear in the subgraphs of more than one gene receive a distinct *multi-gene* colour (dark grey) so that shared interaction hubs are immediately visible. Seed isoforms are highlighted via shape.

All colour palettes are colour-blind-safe (Okabe-Ito).

Value

A `ggplot` object, invisibly. Can be further modified with **ggplot2** layers or saved with `ggplot2::ggsave()`.

A `ggplot` / `patchwork` object, invisibly.

A `ggplot` object, invisibly.

Methods (by class)

- `plot_network(IsoNetAnalyzeR)`: Plot the full network of an IsoNetAnalyzeR session object.
- `plot_network(VirtualPulldownLookup)`: Plot the subgraph of a VirtualPulldownLookup result. The seed node is highlighted in a contrasting colour.
- `plot_network(GeneDiffPulldown)`: Plot a GeneDiffPulldown result.
- `plot_network(DifferentialPulldownAnalysis)`: Plot cross-gene network from a DifferentialPulldownAnalysis result.

Gene colouring

When `group_by_gene = TRUE` and `level = "isoform"`, nodes are coloured by their parent gene. In `"up"`, `"down"`, and `"side_by_side"` modes this replaces the default uniform node colour. In `"overlay"` mode, gene identity is mapped to node *fill* colour while membership (up-only / shared / down-only) is shown via the node *border* (stroke) colour and edge colour, so both dimensions remain visible simultaneously.

Examples

```
obj <- IsoNetAnalyzeR("example")

# Neighbourhood of one isoform
obj <- lookup_virtual_pulldown(obj, "cltcl1_up", seed = "ENST00000621271",
                               order = 1)
p <- plot_network(get_lookup(obj, "cltcl1_up"))

# The same neighbourhood at gene level
p <- plot_network(get_lookup(obj, "cltcl1_up"), level = "gene")

# Transcript identifiers instead of gene symbols, or no labels
p <- plot_network(get_lookup(obj, "cltcl1_up"), node_labels = "id")
p <- plot_network(get_lookup(obj, "cltcl1_up"), node_labels = "none")

seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name == "CLTCL1", ],
  order = 1, confirm = FALSE, verbose = FALSE
)
gr <- get_gene_result(obj, "t2d", "ENSG00000070371")

# Up and down neighbourhoods side by side
p <- plot_network(gr)

# Both neighbourhoods in one overlay
p <- plot_network(gr, direction = "overlay")

seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
result <- get_analysis(obj, "t2d")

# Cross-gene view of the up-labelled seed neighbourhoods
p <- plot_network(result, direction = "up")

# Keep components that share at least one node between genes
p <- plot_network(result, min_shared_nodes = 1)
```

`plot_network_diagnostics` *Plot network diagnostics*

Description

Produces a 2-row x 3-column panel figure summarising the NetworkDiagnostics result stored in an IsoNetAnalyzeR session object. Panels carry no in-plot titles; panel content and the overall figure

title are intended to live in the surrounding caption/body text rather than the image itself.

Usage

```
plot_network_diagnostics(obj, title = NULL)
```

Arguments

obj	An IsoNetAnalyzeR session object with a completed network diagnostics run (see run_network_diagnostics).
title	Character scalar or NULL. Optional overall figure title, off by default since titles are intended to live in the caption rather than the figure itself. Set to a string to override.

Details

Panel layout (2 rows x 3 columns), grouped by network level:

- Row 1 (isoform-level, blue): (A) Isoform degree distribution | (B) Connected component sizes | (C) Clustering coefficient vs degree
- Row 2 (gene-level, orange / categorical): (D) Gene degree distribution | (E) Isoforms per gene | (F) Gene connectivity status

Sticky degree thresholds are shown as dashed vertical lines in the two degree distribution panels (A, D). A LOESS smoothing line is overlaid on the clustering coefficient panel (C). Degree-distribution panels (A, D) and the isoforms-per-gene panel (E) use log-scaled counts to make the heavy right tail readable; no power-law shape is implied by this choice.

Value

A patchwork object (invisibly). Print it to display, or save with ggplot2::ggsave().

Examples

```
# A small deterministic network keeps this example fast; on the bundled
# network the diagnostics take about half a minute.
g <- igraph::make_graph("Zachary")
n <- igraph::vcount(g)
igraph::V(g)$name <- paste0("tx", seq_len(n))
igraph::V(g)$gene <- paste0("gene", ceiling(seq_len(n) / 2))
igraph::V(g)$transcript_biotype <- "protein_coding"
igraph::E(g)$interact <- 1L
toy <- IsoNetAnalyzeR("toy", network = g,
                      metadata = list(network_cutoff = 0.25))
toy <- run_network_diagnostics(toy)
p <- plot_network_diagnostics(toy)
```

print_metadata *Print the metadata of an IsoNetAnalyzeR session object*

Description

Displays every key-value pair stored in the `@metadata` slot in a readable, aligned format. Values that are themselves lists or vectors are shown compactly on one line.

Usage

```
print_metadata(object, ...)

## S4 method for signature 'IsoNetAnalyzeR'
print_metadata(object, ...)
```

Arguments

`object`	An IsoNetAnalyzeR session object.
`...`	Currently unused; reserved for future arguments.

Value

`object`, invisibly.

Methods (by class)

- `print_metadata(IsoNetAnalyzeR)`: Print metadata for an IsoNetAnalyzeR object.

Examples

```
obj <- IsoNetAnalyzeR("example")
obj <- set_metadata(obj, "author", "Jane Doe")
print_metadata(obj)
```

resolve_duplicate_isoforms *Resolve protein-identical isoform identifiers*

Description

Remaps redundant Ensembl transcript identifiers to their kept protein representative. With a network supplied, a mapping is applied only when it is compatible with that graph. Identifiers absent from the map are retained.

Usage

```
resolve_duplicate_isoforms(
  x,
  network = NULL,
  isoform_col = "isoform_id",
  gene_col = "gene_id",
  duplicate_map = NULL
)
```

Arguments

x	Character vector, data.frame or DataFrame containing transcript identifiers.
network	Optional igraph used to validate whether each mapping is applicable.
isoform_col	Name of the transcript-ID column when x is a data frame.
gene_col	Name of the gene-ID column when x is a table.
duplicate_map	Optional map with the schema returned by isonet_duplicate_map(); primarily useful for validated custom maps and tests.

Value

The same type as x, with resolved identifiers and the resolution audit: a dedup_audit attribute, or S4Vectors::metadata(x)$dedup_audit for a DataFrame.

Examples

```
obj <- IsoNetAnalyzeR("example")
map <- isonet_duplicate_map()

# A versioned identifier and a redundant protein-identical transcript,
# resolved against the session network
ids <- c("ENST00000621271.4", map$redundant_ENST[map$network_applicable][1])
resolved <- resolve_duplicate_isoforms(ids, network = get_network(obj))
resolved
attr(resolved, "dedup_audit")
```

---

run_component_analysis

Run a connected-component analysis on a differential pulldown result

---

Description

Decomposes the merged cross-gene network of an existing DifferentialPulldownAnalysis run into its connected components and computes both Tier-1 structural statistics and Tier-2 topological statistics for each component. The result is stored inside the parent analysis and can be accessed with get_component_overview(obj, analysis, label).

Usage

```
run_component_analysis(
  obj,
  analysis,
  label,
  direction = c("both", "up", "down"),
  level = c("isoform", "gene"),
  genes = NULL,
  min_component_nodes = NULL,
  min_shared_nodes = NULL,
  min_edges = NULL
)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of an existing DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
label	Character scalar. Key under which to store the result.
direction	`"both"` (default), `"up"`, or `"down"`.
level	`"isoform"` (default) or `"gene"`.
genes	Character vector of gene IDs to include, or `NULL` for all genes.
min_component_nodes	Integer or `NULL`. Drop components with fewer nodes than this.
min_shared_nodes	Integer or `NULL`. Drop components with fewer shared (multi-gene) nodes than this.
min_edges	Integer or `NULL`. Drop components with fewer edges than this.

Value

An updated IsoNetAnalyzeR object (invisibly). Access results with `get_component_overview` (combined Tier-1 and Tier-2 table) and `get_component` (per-node table for a single component).

Tier-1 statistics (structural)

- **Size**: nodes, edges, density, average degree, clustering coefficient.
- **Seed enrichment**: count and fraction of seed vs. neighbour nodes.
- **Direction mixing**: up/down seed counts and contributing genes; the `mixed` flag records components containing both labels.
- **Multi-gene hubs**: count and fraction of shared nodes.

Tier-2 statistics (topological)

- **Diameter**: longest shortest path in the component. Large diameter suggests a signalling cascade rather than a tight complex.
- **Radius**: minimum eccentricity across all nodes.
- **Degree centralization**: how unequally degree is distributed (0 = perfectly uniform, 1 = pure star graph).

- **Topology class**: heuristic classification into `"hub-spoke"`, `"chain-like"`, or `"clique-like"` based on degree centralization (see ComponentAnalysis Details).
- **Hub betweenness**: for every shared (multi-gene) node, betweenness centrality computed within its component, normalised to [0, 1], and ranked. Nodes with high normalised betweenness are structural bridges across the sub-communities pulled in by different genes. Biological relevance requires external evidence.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)

# Isoform-level components of both seed directions
obj <- run_component_analysis(obj, "t2d", label = "comp")
overview <- get_component_overview(obj, "t2d", "comp")

# Components containing both up and down seed labels
overview[overview$mixed, c("component_id", "n_nodes", "n_shared_nodes")]

# Gene-level components
obj <- run_component_analysis(obj, "t2d", label = "comp_gene",
                              level = "gene")
```

---

run_component_ora *Run GO over-representation analysis on connected components*

Description

Tests each connected component from a ComponentAnalysis run against GO gene sets (or a user-supplied pathway list) using `fgsea::fora()`. The result annotates each component with the GO terms most significantly over-represented among its member genes, providing a biological interpretation of what each protein community is doing.

Usage

```
run_component_ora(
  obj,
  analysis,
  component_analysis,
  label,
  pathways = NULL,
  go_category = c("GO:BP", "GO:MF", "GO:CC", "ALL"),
  nodes = c("all", "seeds"),
  universe = NULL,
  minSize = 10L,
  maxSize = 500L
)
```

Arguments

obj	An IsoNetAnalyzeR session object.
analysis	Character scalar. Label of the parent DifferentialPulldownAnalysis.
component_analysis	Character scalar. Label of the ComponentAnalysis to annotate.
label	Character scalar. Key under which the result is stored; retrieve it afterwards with `get_component_ora_result(obj, analysis, component_analysis, label)`.
pathways	Named list of character vectors (Ensembl gene IDs per pathway), a `GeneSetCollection`, or `NULL` (default) to fetch MSigDB GO gene sets with the msigdb package.
go_category	Character scalar. GO subcollection to fetch when `pathways = NULL`. One of `"GO:BP"`, `"GO:MF"`, `"GO:CC"`, or `"ALL"`. Default `"GO:BP"`.
nodes	Character scalar. Which nodes to test per component: `"all"` (default) or `"seeds"`.
universe	Character vector of Ensembl gene IDs to use as the ORA background, or `NULL` (default) to use all genes in the IsoNetAnalyzeR network.
minSize	Integer. Minimum gene set size after filtering to the universe. Default `10`.
maxSize	Integer. Maximum gene set size. Default `500`.

Value

The IsoNetAnalyzeR session object, invisibly, with the ComponentOraResult stored under `label` inside the component analysis. Access via `get_component_ora_result(obj, analysis, component_analysis, label)`.

Node-to-gene mapping

Nodes in the component network are isoform IDs. The reliable `gene` vertex attribute on the IsoNetAnalyzeR network object is used to map each isoform to its Ensembl gene ID. This mapping is applied automatically; no user input is required.

Background universe

The default background is all genes represented in the full IsoNetAnalyzeR network. This is the most principled choice for protein-interaction network ORA because it restricts the background to proteins that could in principle have appeared in any component. Using all GO-annotated genes would inflate apparent enrichment by including thousands of proteins absent from the network. You can override this with the `universe` argument.

GO pathways

`pathways` may be a named list of Ensembl gene ID vectors or a GSEABase `GeneSetCollection`, for example from the msigdb package; gene symbols are mapped to Ensembl gene IDs with the Ensembl 107 annotation (`isonet_gene_names`). When `pathways = NULL` (default), the GO gene sets of MSigDB release 2023.1 (collection C5) are fetched with the msigdb package from ExperimentHub. `go_category` selects the subcollection:

`"GO:BP"` Biological process - *what* the proteins do. Most informative for isoform-switch interpretation. **Default.**

`"GO:MF"` Molecular function - the biochemical activity. Useful for identifying domain-level functional differences between isoforms.

`"GO:CC"` Cellular component - subcellular location. Less directly relevant to isoform switching; use as secondary context.

`"ALL"` All three GO subcollections combined (excludes HPO).

Which nodes to test

`nodes = "all"` **(default)** All nodes in each component are tested. Recommended when components are small (typical for differential pulldown results), as using all nodes maximises the gene set size and statistical power.

`nodes = "seeds"` Only seed nodes (the experimentally regulated isoforms) are tested. More conservative; useful when components are large and neighbours dominate.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
obj <- run_component_analysis(obj, "t2d", label = "comp")
# Two small gene sets of gene symbols, for illustration. MSigDB collections
# are available as GeneSetCollections from the msigdb package; see the
# vignette.
gene_sets <- GSEABase::GeneSetCollection(list(
  GSEABase::GeneSet(c("WAS", "VAV1", "ICOS"), setName = "set_A",
                    geneIdType = GSEABase::SymbolIdentifier()),
  GSEABase::GeneSet(c("CLTCL1", "PRKACB", "HLA-DMA"), setName = "set_B",
                    geneIdType = GSEABase::SymbolIdentifier())
))
obj <- run_component_ora(obj, "t2d", "comp", label = "example",
                         pathways = gene_sets, minSize = 2)
get_component_ora_result(obj, "t2d", "comp", "example")

# Most significant gene set per component
get_ora_table(obj, "t2d", "comp", "example", top_n = 1)
```

---

run_differential_pulldown

Run a differential virtual pulldown analysis

Description

Reads a long-format table of caller-selected isoform seeds (one isoform per row), runs a virtual pulldown for the up- and down-labelled seeds of each gene separately, computes per-direction and per-gene statistics, and adds a DifferentialPulldownAnalysis object to the IsoNetAnalyzeR session object.

Usage

```
run_differential_pulldown(
  obj,
  label,
  input,
```

```
  order,
  cutoff = NULL,
  parallel = FALSE,
  confirm = interactive(),
  verbose = TRUE,
  resolve_duplicates = TRUE,
  seed_padj_col = NULL,
  seed_fdr = 0.05
)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`label`	Character scalar. Key under which to store the result; retrieve it afterwards with `get_analysis(obj, label)`.
`input`	One of: • A character scalar - path to a CSV or Excel file. • A `data.frame` or S4Vectors `DataFrame`.
`order`	Integer scalar. Number of BFS hops for the virtual pulldown.
`cutoff`	Numeric scalar in [0, 1]. Internal-edge fraction cutoff. Defaults to `obj@metadata[["network_cutof` when `NULL`.
`parallel`	Logical. When `TRUE`, per-gene pulldowns are dispatched via `future.apply::future_lapply()` using the active **future** plan. Requires the **future** and **future.apply** packages to be installed. Default `FALSE`.
`confirm`	Logical. When `TRUE` (the default in interactive sessions), a network coverage report is printed before the pulldowns begin and the user is asked to confirm before proceeding. Set to `FALSE` to skip the prompt in scripts or non-interactive contexts. The coverage report is always printed regardless of this flag; only the interactive pause is suppressed.
`verbose`	Logical. Print per-gene progress messages. Default `TRUE`.
`resolve_duplicates`	Logical. Resolve versioned and protein-identical redundant transcript identifiers before analysis. Same-direction representative collisions are collapsed and genes with opposite-direction collisions are excluded. Default `TRUE`; use `FALSE` to match identifiers exactly.
`seed_padj_col`	Optional name of the caller-supplied adjusted p-value column. When non-NULL, enforce significant retained support and both mapped directions after biological eligibility and identifier resolution.
`seed_fdr`	Numeric scalar in (0, 1]. A retained seed is significant when its supplied adjusted p-value is strictly below this threshold. Default 0.05. No additional p-value adjustment is performed.

Value

An updated IsoNetAnalyzeR object (invisibly); access the result via `get_analysis(obj, label)`.

Input table format

A CSV, Excel (`.xlsx` / `.xls`), `data.frame` or `DataFrame` in **long format** with exactly the following columns (any order):

gene_id Gene identifier matching the network's gene vertex attribute; an Ensembl gene ID for the bundled network. A version suffix (`ENSG00000070371.16`) is removed unless the network uses the versioned ID.

isoform_id Isoform / transcript identifier matching vertex names in the session network.

direction Must be exactly `"up"` or `"down"` (case insensitive; trimmed).

Extra columns are retained in `@input_table`. If `seed_padj_col` is supplied, that column is used for retained-seed validation.

Retained-seed support

With `seed_padj_col` supplied, decay filtering and identifier resolution occur first, then unmapped rows are removed. Each retained gene must have both up and down seeds and at least one retained adjusted p-value below `seed_fdr`. Other retained seeds may be non-significant or have missing adjusted p-values. No maximum seed count is imposed. Supply only transcripts passing the study's gene and effect-size filters; this function does not perform RNA testing or choose an effect floor. Audits are stored in `obj@metadata$analysis_seed_support[[label]]`. With `seed_padj_col = NULL`, the generic three-column exploratory contract is unchanged and does not assert statistical support.

Parallel execution

Parallelism is opt-in and requires the **future** and **future.apply** packages (`install.packages(c("future", "future.apply"))`). When `parallel = TRUE` the per-gene pulldowns are dispatched via `future.apply::future_lapp` which delegates all scheduling to whatever **future** *plan* the user has set before calling this function.

Typical setup:

```
library(future)

# Recommended: works on all platforms including Windows
plan(multisession, workers = 4)

# Unix/macOS only: lower overhead via forking
plan(multicore, workers = 4)

# Back to sequential when done
plan(sequential)
```

If no plan has been set explicitly, **future** defaults to `sequential`, so passing `parallel = TRUE` without configuring a plan is safe but will not speed anything up. An informative message is emitted in that situation.

The network graph embedded in `obj` is treated as read-only inside each worker; no mutable state is shared between workers.

Progress reporting

A progress bar is always shown during the per-gene loop via **progressr**. The bar works in both sequential and parallel modes. Users can customise the handler before calling this function, e.g.:

```
progressr::handlers("cli")          # rich bar (default when cli is installed)
progressr::handlers("txtprogressbar") # plain fallback
```

If `progressr::handlers(global = TRUE)` has been set in the session, that handler is used automatically; otherwise `with_progress()` applies the default handler locally for this call only.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
cltcl1 <- seeds[seeds$gene_name == "CLTCL1", ]
obj <- IsoNetAnalyzeR("example")

# Exploratory run: gene_id, isoform_id and direction only
obj <- run_differential_pulldown(obj, "explore", cltcl1, order = 1,
                                 confirm = FALSE, verbose = FALSE)

# Require at least one significant retained seed per gene
obj <- run_differential_pulldown(
  obj, "supported", cltcl1, order = 1, confirm = FALSE, verbose = FALSE,
  seed_padj_col = "within_gene_padj", seed_fdr = 0.05
)
get_metadata(obj, "analysis_seed_support")$supported$gene_audit
get_cross_gene_summary(obj, "supported")
```

`run_fgsea` *Run GSEA on a differential pulldown analysis*

Description

Ranks genes by a network-topology delta derived from the `cross_gene_summary` table of an existing DifferentialPulldownAnalysis run, then tests the ranked list against GO gene sets (or a user-supplied pathway list) using `fgsea::fgsea()`.

Usage

```
run_fgsea(
  obj,
  analysis,
  label,
  rank_by = "delta_nodes",
  stats = NULL,
  pathways = NULL,
  go_category = c("GO:BP", "GO:MF", "GO:CC", "ALL"),
  minSize = 15L,
  maxSize = 500L,
  nperm = NULL,
  ...
)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`analysis`	Character scalar. Label of the DifferentialPulldownAnalysis; use `list_analyses(obj)` to see available labels.
`label`	Character scalar. Key under which the result will be stored; retrieve it afterwards with `get_gsea_result(obj, analysis, label)`.

rank_by	Character scalar. One of the built-in ranking presets (see Ranking tiers above). Ignored when stats is supplied. Default "delta_nodes".
stats	Named numeric vector or NULL. When supplied, this vector is used directly as the fgsea rank statistic, bypassing rank_by. Names must be gene IDs. Default NULL.
pathways	Named list of character vectors (Ensembl gene IDs per pathway), a GeneSetCollection, or NULL (default) to fetch MSigDB GO gene sets with the msigdb package (see Gene sets).
go_category	Character scalar. GO subcollection to fetch when pathways = NULL. One of "GO:BP", "GO:MF", "GO:CC", or "ALL". Default "GO:BP".
minSize	Integer. Minimum gene set size after filtering to genes present in the ranked list. Default 15.
maxSize	Integer. Maximum gene set size. Default 500.
nperm	Integer or NULL (default). With NULL, fgsea estimates p-values with fgseaMultilevel, which also estimates very small p-values (down to its eps, which can be passed through ...). A number runs fgseaSimple with that many permutations, which cannot give a p-value below 1 / (nperm + 1).
...	Additional arguments passed to fgsea::fgsea().

Value

The IsoNetAnalyzeR session object, invisibly, with the FgseaResult stored under label inside the analysis. Access via get_gsea_result(obj, analysis, label).

Ranking tiers

Tier 1 - built-in preset (rank_by) A named delta computed as the up-seed metric minus the down-seed metric from cross_gene_summary. Positive values indicate a larger static predicted neighbourhood metric for the up-labelled seeds; negative values indicate the opposite. Node, edge and gene counts include the seeds themselves, so a direction with more seeds counts more; compare n_up_resolved and n_down_resolved.

```
Available presets:
\itemize{
  \item \code{"delta_nodes"} - up_nodes \eqn{-} down_nodes \strong{(default)}
   \item \code{"delta_edges"} - up_edges \eqn{-} down_edges
   \item \code{"delta_avg_degree"} - up_avg_degree \eqn{-} down_avg_degree
   \item \code{"delta_density"} - up_density \eqn{-} down_density
   \item \code{"delta_clustering"} - up_clustering \eqn{-} down_clustering
   \item \code{"delta_starness"} - up_starness \eqn{-} down_starness
   \item \code{"delta_genes"} - up_genes \eqn{-} down_genes
  \item \code{"delta_seed_mean_pi"} - up_seed_mean_pi \eqn{-} down_seed_mean_pi
  \item \code{"delta_seed_pi_entropy"} - up_seed_pi_entropy \eqn{-} down_seed_pi_entropy
   \item \code{"edges_gained_vs_lost"} - n_edges_gained \eqn{-} n_edges_lost
}
```

Tier 3 - full override (stats) Pass a named numeric vector directly (names must be gene IDs matching the pathway gene sets). This completely bypasses the built-in ranking logic and rank_by is ignored. Useful for complex custom rankings:

```
\preformatted{
    cgs      <- get_cross_gene_summary(obj, "my_run")
```

```
my_stats <- setNames(
  (cgs$up_nodes - cgs$down_nodes) * 0.7 +
  (cgs$up_edges - cgs$down_edges) * 0.3,
  cgs$gene_id
)
obj <- run_fgsea(obj, "my_run", label = "weighted", stats = my_stats)
}
```

Gene sets

`pathways` may be a named list of Ensembl gene ID vectors or a GSEABase `GeneSetCollection`, for example a collection from the msigdb package. Gene symbols in a `GeneSetCollection` are mapped to Ensembl gene IDs with the Ensembl 107 annotation (`isonet_gene_names`). The gene IDs of a collection without an identifier type, such as one read with `GSEABase::getGmt()`, are taken as Ensembl gene IDs when they all look like Ensembl gene IDs, and as gene symbols otherwise. Gene sets are matched to the IDs of the ranked genes (the analysed gene IDs, or the names of `stats`): IDs that are ranked genes are kept, and other Ensembl gene IDs match the ranked genes with or without a version suffix. Gene sets with no ranked gene are an error.

When `pathways = NULL` (default), the GO gene sets of MSigDB release 2023.1 (collection C5) are fetched with the msigdb package from ExperimentHub and kept for the R session. `go_category` selects the subcollection(s): `"GO:BP"` (biological process), `"GO:MF"` (molecular function), `"GO:CC"` (cellular component), or `"ALL"` for all three. fgsea estimates p-values by random sampling, so set the random seed before this call.

Examples

```
seeds <- read.csv(system.file("extdata", "gse141432_t2d_vs_lean_seeds.csv",
                              package = "IsoNetAnalyzeR"))
# Six genes for speed; the vignette analyses all 65. With few ranked genes,
# minSize must be small.
genes <- c("CLTCL1", "WAS", "VAV1", "HLA-DMA", "ICOS", "PRKACB")
obj <- IsoNetAnalyzeR("example")
obj <- run_differential_pulldown(
  obj, "t2d", seeds[seeds$gene_name %in% genes, ],
  order = 1, confirm = FALSE, verbose = FALSE
)
# Two small gene sets of gene symbols, for illustration. MSigDB collections
# are available as GeneSetCollections from the msigdb package; see the
# vignette.
gene_sets <- GSEABase::GeneSetCollection(list(
  GSEABase::GeneSet(c("WAS", "VAV1", "ICOS"), setName = "set_A",
                    geneIdType = GSEABase::SymbolIdentifier()),
  GSEABase::GeneSet(c("CLTCL1", "PRKACB", "HLA-DMA"), setName = "set_B",
                    geneIdType = GSEABase::SymbolIdentifier())
))
set.seed(42)
obj <- run_fgsea(obj, "t2d", label = "example_gsea", pathways = gene_sets,
                 minSize = 3)
get_gsea_table(obj, "t2d", "example_gsea")

# Custom ranking: a weighted node and edge difference per gene
cgs <- get_cross_gene_summary(obj, "t2d")
my_stats <- setNames(
  (cgs$up_nodes - cgs$down_nodes) * 0.7 +
    (cgs$up_edges - cgs$down_edges) * 0.3,
```

```
  cgs$gene_id
)
obj <- run_fgsea(obj, "t2d", label = "weighted", stats = my_stats,
                 pathways = gene_sets, minSize = 3)
```

`run_network_diagnostics` *Run a comprehensive network diagnostics analysis*

Description

Characterises the isoform interaction network loaded in an IsoNetAnalyzeR session object. Computes:

- Structural summary statistics across three network views (full, predicted isoform, gene-level predicted).
- Global topology metrics: density, average degree, clustering coefficient, connected components, diameter, and average path length of the largest component.
- Per-isoform and per-gene degree tables.
- Connected component size table.
- Clustering coefficient vs degree table.
- Per-gene isoform connectivity status (`"all_connected"`, `"partial"`, `"all_isolated"`).
- Sticky isoform and gene tables based on a degree threshold.

Usage

```
run_network_diagnostics(
  obj,
  sticky_method = c("quantile", "sd"),
  sticky_cutoff = NULL
)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`sticky_method`	Character scalar. Threshold method for sticky node detection: `"quantile"` (default) or `"sd"`.
`sticky_cutoff`	Numeric scalar. For `"quantile"`: a probability in [0, 1] (default `0.95`). For `"sd"`: a positive multiplier (default `2`).

Details

Results are stored in `obj@network_diagnostics` as a NetworkDiagnostics object and can be retrieved with `get_network_diagnostics`. Re-running overwrites any previous result with a warning.

Value

The updated IsoNetAnalyzeR object, invisibly. Access the result with `get_network_diagnostics(obj)`.

Sticky node detection

An isoform (or gene) is classified as sticky if its degree in the predicted subnetwork exceeds a threshold derived from the overall degree distribution. Two methods are supported:

`"quantile"` **(default)** Threshold is the `sticky_cutoff` quantile of the degree distribution. `sticky_cutoff = 0.95` means the top 5% by degree are sticky. Robust to the heavy-tailed distributions typical of protein interaction networks.

`"sd"` Threshold is `mean + sticky_cutoff * SD`. More sensitive to extreme hubs inflating the mean.

Examples

```
# A small deterministic network keeps this example fast; on the bundled
# network the diagnostics take about half a minute.
g <- igraph::make_graph("Zachary")
n <- igraph::vcount(g)
igraph::V(g)$name <- paste0("tx", seq_len(n))
igraph::V(g)$gene <- paste0("gene", ceiling(seq_len(n) / 2))
igraph::V(g)$transcript_biotype <- "protein_coding"
igraph::E(g)$interact <- 1L
toy <- IsoNetAnalyzeR("toy", network = g,
                      metadata = list(network_cutoff = 0.25))

# Default: sticky nodes are the top 5% by degree
toy <- run_network_diagnostics(toy)
toy

# Standard-deviation rule instead of a quantile
toy <- run_network_diagnostics(toy, sticky_method = "sd", sticky_cutoff = 2)
```

`set_metadata` *Set a metadata field on an IsoNetAnalyzeR session object*

Description

Adds or updates a single named entry in the `@metadata` slot of an IsoNetAnalyzeR session object. If the key already exists its value is silently overwritten; if it does not exist it is created.

Usage

```
set_metadata(obj, key, value)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`key`	Character scalar. The metadata field name.
`value`	Any R object to store under `key`.

Details

The following keys are set automatically by the constructor and should not normally be overwritten: `network_source`, `pi_threshold`, `network_cutoff`.

Value

An updated IsoNetAnalyzeR object (invisibly).

Examples

```
obj <- IsoNetAnalyzeR("example")
obj <- set_metadata(obj, "author", "Jane Doe")
obj <- set_metadata(obj, "notes", "Pilot run")
get_metadata(obj, "author")
```

---

summarize_subgraphs *Summarize a list of virtual pulldown subgraphs*

Description

Computes a set of network-level statistics for each subgraph in `vp_list`, returning one row per subgraph. When a subgraph has multiple seed vertices, their names are collapsed into a single comma-separated string.

Usage

```
summarize_subgraphs(vp_list)
```

Arguments

`vp_list`	A list of igraph objects, typically the output of repeated calls to `virtual_pulldown`. Names are optional.

Value

A `DataFrame` with one row per subgraph and the following columns:

seed Comma-separated seed isoform name(s).

nodes Total number of vertices.

edges Total number of edges.

density Edge density of the subgraph.

avg_degree Mean vertex degree.

clustering Average local clustering coefficient.

starness Degree centralization score (0 = uniform, 1 = perfect star).

genes Number of unique genes represented.

Examples

```
obj <- IsoNetAnalyzeR("example")
neighbourhoods <- list(
  cltcl1_up = virtual_pulldown(obj, "ENST00000621271", order = 1),
  cltcl1_down = virtual_pulldown(obj, "ENST00000427926", order = 1)
)
summarize_subgraphs(neighbourhoods)
```

`VirtualPulldownLookup-class` *VirtualPulldownLookup - single quick virtual pulldown look-up*

Description

Stores the result of a one-shot virtual pulldown query at either isoform or gene level, without running a full analysis across many seeds. Can optionally be stored inside an IsoNetAnalyzeR session object.

Usage

```
## S4 method for signature 'VirtualPulldownLookup'
show(object)
```

Arguments

`object`	An VirtualPulldownLookup session object.

Value

Objects of this class are created by `lookup_virtual_pulldown` and returned by `get_lookup`.

Methods (by generic)

- `show(VirtualPulldownLookup)`: Print a concise summary.

Slots

`seed` Character vector. The seed isoform(s) or gene(s) queried. Multiple seeds are stored as a vector when all were used as order-0 anchors in a single pulldown.

`level` Character scalar. Either `"isoform"` or `"gene"`.

`subgraph` An `igraph` object representing the pulldown subnetwork.

`node_stats` `DataFrame`. Per-node statistics for the subgraph (output of `analyze_subgraph`).

`cutoff` Numeric scalar. The internal-edge fraction cutoff used for this look-up. Stored explicitly so it is always visible on the object independent of the session metadata.

`params` Named list. Additional parameters used: `order`, `level`.

`created_at` `POSIXct` scalar. Timestamp of the look-up.

Examples

```
obj <- IsoNetAnalyzeR("example")
obj <- lookup_virtual_pulldown(obj, "cltcl1_up", seed = "ENST00000621271",
                               order = 1)
get_lookup(obj, "cltcl1_up")
```

`virtual_pulldown` *Virtual pulldown*

Description

Performs a virtual pulldown by first restricting the network to predicted interaction edges (`interact == 1`), expanding from a set of seed isoforms up to `order` hops, then filtering out non-specific nodes whose edges are predominantly external to the subnetwork.

Usage

```
virtual_pulldown(
  obj,
  seed_vertices,
  order,
  cutoff = NULL,
  resolve_duplicates = TRUE
)
```

Arguments

`obj`	An IsoNetAnalyzeR session object.
`seed_vertices`	Character vector of seed isoform vertex names. All supplied vertices are treated as order-0 anchors.
`order`	Integer. Number of interaction hops to expand from the seeds.
`cutoff`	Numeric scalar in [0, 1]. Minimum fraction of edges that must be internal to the subnetwork for a node to be retained. When `NULL` (default) the value stored in `obj@metadata[["network_cutoff"]]` is used.
`resolve_duplicates`	Logical. Resolve versioned and protein-identical redundant transcript IDs against the active network. Default `TRUE`.

Details

The network is taken directly from the IsoNetAnalyzeR session object. The cutoff defaults to the value stored in `obj@metadata[["network_cutoff"]]` but can be overridden to explore alternative thresholds.

Value

An igraph object representing the filtered virtual pulldown subnetwork. Vertices carry an `order` attribute (hop distance from the nearest seed) and a `frac_internal` attribute (fraction of their interaction edges that remain internal to the subnetwork). Seed vertices with zero `interact == 1` edges are retained as isolated nodes (`frac_internal = NA`, `degree = 0`), representing isoforms present in the pinned network but incident to no predicted edge in that network.

Examples

```
obj <- IsoNetAnalyzeR("example")
g <- virtual_pulldown(obj, "ENST00000621271", order = 1)
c(nodes = igraph::vcount(g), edges = igraph::ecount(g))
```

Index

aggregate_duplicate_counts, 3
analyze_subgraph, 4, *12*, *23, 24*, *67*

ComponentAnalysis, *5*, *7*, *9*, *13*, *15, 16*, *28*, *39, 40*, *56, 57*
ComponentAnalysis-class, 5
ComponentOraResult, *6, 7*, *15*, *27, 28*, *57*
ComponentOraResult-class, 7

DifferentialPulldownAnalysis, *5–8*, *10, 11*, *13*, *15–19*, *21, 22*, *28*, *32*, *39*, *42, 43*, *45–47*, *49, 50*, *54, 55*, *57, 58*, *61*
DifferentialPulldownAnalysis-class, 8

FgseaResult, *9, 10*, *18–21*, *62*
FgseaResult-class, 10

GeneDiffPulldown, *8, 9*, *11*, *17, 18*, *29*, *47*, *50*
GeneDiffPulldown-class, 11
get_analysis, *9*, 12
get_analysis,IsoNetAnalyzeR-method *(*IsoNetAnalyzeR-class*)*, 31
get_component, *6*, 13, *16*, *55*
get_component_ora_result, *7*, 14
get_component_ora_result,IsoNetAnalyzeR-method *(*get_component_ora_result*)*, 14
get_component_overview, *6*, *13*, 15, *55*
get_cross_gene_summary, 16
get_cross_gene_summary,IsoNetAnalyzeR-method *(*get_cross_gene_summary*)*, 16
get_gene_result, *12*, 17, *29*
get_gene_result,IsoNetAnalyzeR-method *(*get_gene_result*)*, 17
get_gsea_result, *10*, 18
get_gsea_result,IsoNetAnalyzeR-method *(*get_gsea_result*)*, 18
get_gsea_stats, 19
get_gsea_stats,IsoNetAnalyzeR-method *(*get_gsea_stats*)*, 19
get_gsea_table, 20
get_gsea_table,IsoNetAnalyzeR-method *(*get_gsea_table*)*, 20
get_isoform_stats, 22
get_isoform_stats,IsoNetAnalyzeR-method *(*get_isoform_stats*)*, 22
get_lookup, 23, *67*
get_lookup,IsoNetAnalyzeR-method *(*IsoNetAnalyzeR-class*)*, 31
get_lookup_stats, 23
get_lookup_stats,IsoNetAnalyzeR-method *(*get_lookup_stats*)*, 23
get_lookup_subgraph, 24
get_lookup_subgraph,IsoNetAnalyzeR-method *(*get_lookup_subgraph*)*, 24
get_metadata, 25
get_network, 26
get_network_diagnostics, 26, *37*, *64*
get_network_diagnostics,IsoNetAnalyzeR-method *(*get_network_diagnostics*)*, 26
get_ora_table, 27
get_ora_table,IsoNetAnalyzeR-method *(*get_ora_table*)*, 27
get_pulldown_graph, 29
get_pulldown_graph,GeneDiffPulldown-method *(*get_pulldown_graph*)*, 29

isonet_duplicate_map, 33
isonet_gene_names, 33, *49*, *57*, *63*
isonet_transcript_biotypes, *33*, 34
IsoNetAnalyzeR, *13*, *15–28*, *30*, 30, *31, 32*, *35–37*, *39*, *42, 43*, *45–47*, *50–53*, *55*, *57–59*, *61, 62*, *64–68*
IsoNetAnalyzeR-class, 31

list_analyses, 35
list_analyses,IsoNetAnalyzeR-method *(*IsoNetAnalyzeR-class*)*, 31
list_lookups, 35
list_lookups,IsoNetAnalyzeR-method *(*IsoNetAnalyzeR-class*)*, 31
lookup_virtual_pulldown, 36, *67*

NetworkDiagnostics, *26*, *32*, *37*, *51*, *64*
NetworkDiagnostics-class, 37

plot_component, 39
plot_cross_gene_boxplot, 41, *43*
plot_cross_gene_boxplot_overview, 43
plot_cross_gene_overview, 44, *45*

plot_cross_gene_scatter, *44*, 45
plot_network, *39, 40*, 47
plot_network,DifferentialPulldownAnalysis-method
(plot_network), 47
plot_network,GeneDiffPulldown-method
(plot_network), 47
plot_network,IsoNetAnalyzeR-method
(plot_network), 47
plot_network,VirtualPulldownLookup-method
(plot_network), 47
plot_network_diagnostics, 51
print_metadata, *25*, 53
print_metadata,IsoNetAnalyzeR-method
(print_metadata), 53

resolve_duplicate_isoforms, 53
run_component_analysis, *5, 6*, *9*, 54
run_component_ora, *6, 7*, 56
run_differential_pulldown, *8, 9*, 58
run_fgsea, *9, 10*, 61
run_network_diagnostics, *26*, *32*, *37*, *52*,
64

set_metadata, *25*, 65
show,ComponentAnalysis-method
(ComponentAnalysis-class), 5
show,ComponentOraResult-method
(ComponentOraResult-class), 7
show,DifferentialPulldownAnalysis-method
(DifferentialPulldownAnalysis-class),
8
show,FgseaResult-method
(FgseaResult-class), 10
show,GeneDiffPulldown-method
(GeneDiffPulldown-class), 11
show,IsoNetAnalyzeR-method
(IsoNetAnalyzeR-class), 31
show,NetworkDiagnostics-method
(NetworkDiagnostics-class), 37
show,VirtualPulldownLookup-method
(VirtualPulldownLookup-class),
67
summarize_subgraphs, 66

virtual_pulldown, *66*, 68
VirtualPulldownLookup, *23, 24*, *32*, *36*, *47*,
50, *67*
VirtualPulldownLookup-class, 67