Package: Fancy (via r-universe)

November 25, 2026

Type Package

Title Frequency and Nonlinear Correlation Hybrid Network Inference

Version 1.0.0

URL https://github.com/wala-github/Fancy

BugReports https://github.com/wala-github/Fancy/issues

Description Infers microbial co-abundance networks by combining mutual information (via MRNET) and distance correlation into a single hybrid edge score. Bootstrap resampling provides frequency-based confidence, and a weighted scoring function ranks edges for downstream module analysis. Designed for metagenome-assembled genome (MAG) count tables.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.5.0)

Imports minet, compositions, energy, knnmi, snowfall, methods, dplyr, tidyr, purrr, readr, rlang, tibble, stringr, graphics, stats, utils

Suggests BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0), ggplot2, igraph, magick

biocViews NetworkInference, Network, Metagenomics, Microbiome, StatisticalMethod

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libgsl0-dev libicu-dev libx11-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:56:46 UTC

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

RemoteRef RELEASE_3_24

RemoteSha 7657729684bb0a701d8b71e22c0993c761e58fa3

Contents

$,FancyResult-method . . . 2
bootstrap_networks . . . 3
clean_sample_names . . . 4
clr_normalize . . . 5
export_cytoscape . . . 5
fancy . . . 6
fancy_tiny_clr . . . 8
fancy_tiny_counts . . . 9
fancy_tiny_coverage . . . 10
fancy_tiny_metadata . . . 10
fancy_tiny_taxonomy . . . 11
FancyResult-class . . . 12
filter_mags . . . 12
find_optimal_k . . . 13
hybrid_score . . . 14
parse_count_tables . . . 16
parse_gtdb_taxonomy . . . 17
phyla_palette . . . 18
plot,FancyResult,missing-method . . . 19
plot_k_elbow . . . 20
plot_network . . . 20
show,FancyResult-method . . . 21
threshold_edges . . . 22

Index **23**

`$,FancyResult-method` *Access FancyResult slots with* `$`

Description

Convenience accessor so that `result$edges`, `result$all_edges`, `result$params`, etc. retrieve the corresponding slot of a FancyResult object.

Usage

```
## S4 method for signature 'FancyResult'
x$name
```

Arguments

x	A FancyResult object.
name	Slot name to extract.

Value

The contents of the named slot.

---

bootstrap_networks *Bootstrap network inference*

Description

Runs n_bootstrap bootstrap iterations in parallel using **snowfall**. Each iteration resamples rows, computes MI and dCor networks, and returns edge lists. Results are collected into a single named list.

Usage

```
bootstrap_networks(
  data,
  n_bootstrap = 100L,
  k = 5L,
  cpus = 1L,
  use_mrnet = TRUE,
  save_path = NULL
)
```

Arguments

data	A numeric matrix or data.frame with observations (samples) as rows and variables (MAGs) as columns.
n_bootstrap	Integer; number of bootstrap iterations.
k	Integer; kNN parameter for compute_mi_matrix().
cpus	Integer; number of parallel workers. For large datasets (>1000 MAGs), consider using more CPUs (e.g. cpus = 14L).
use_mrnet	Logical; if TRUE (default), applies MRNET to the MI matrix for sparse edge selection. If FALSE, uses all pairwise MI edges directly.
save_path	Optional directory path. When non-NULL, each bootstrap result is saved as an RDS file for recovery.

Value

A named list with elements MRNET_1, dCor_1, MRNET_2, dCor_2, etc.

Examples

```
small_mat <- matrix(rnorm(50),
    nrow = 10, ncol = 5,
    dimnames = list(NULL, paste0("V", 1:5))
)
res <- bootstrap_networks(small_mat, n_bootstrap = 2, k = 2L, cpus = 1)
names(res)[1:4]
```

`clean_sample_names` *Clean sample names from TSV column headers*

Description

Removes the `X` prefix that `utils::read.csv()` adds to column names starting with a digit, replaces `.` with `-` to restore the original sample IDs, and strips format-specific suffixes such as `.QUALITY_PASSED_R1.fastq.*`.

Usage

```
clean_sample_names(
  x,
  remove_prefix = TRUE,
  dot_to_dash = TRUE,
  suffix_patterns = c("\\.QUALITY_PASSED_R1\\.fastq\\.Read\\.Count$",
    "\\.QUALITY_PASSED_R1\\.fastq\\.Covered\\.Fraction$",
    "\\.QUALITY_PASSED_R1\\.fastq\\.Relative\\.Abundance.*$")
)
```

Arguments

`x`	A character vector of sample names (typically `colnames(df)`).
`remove_prefix`	Logical; remove leading `X` added by R? Default `TRUE`.
`dot_to_dash`	Logical; replace `.` with `-`? Default `TRUE`.
`suffix_patterns`	Character vector of regex patterns to strip from the end of each name.

Value

A character vector of cleaned names.

Examples

```
clean_sample_names(c("X12.34.AB.QUALITY_PASSED_R1.fastq.Read.Count"))
```

clr_normalize *CLR-normalise a count table*

Description

Applies centred log-ratio (CLR) transformation via `compositions::clr()`, adding a pseudocount first to handle zeros. Equivalent to the original `clr(as.matrix(selected_count + 1))`.

Usage

```
clr_normalize(count_table, pseudocount = 1)
```

Arguments

`count_table`	A data.frame or matrix of counts (MAGs as rows, samples as columns).
`pseudocount`	Numeric value added to all counts before transformation. Default `1`.

Value

A data.frame of CLR-transformed values preserving the original row and column names.

Note

In the original `02_Preprocessing.Rmd` the CLR output was saved as `normalised_count`, but every downstream script (05_, 06_, 07_) references `clr_transform_df`. The saved RData was renamed at some point; this function's output corresponds to both names.

Examples

```
data(fancy_tiny_counts)
clr_df <- clr_normalize(fancy_tiny_counts[1:10, 1:10])
dim(clr_df)
```

export_cytoscape *Export a fancy result for Cytoscape*

Description

Writes node and edge tables as tab-separated files suitable for import into Cytoscape. The node table includes taxonomy columns and a hex colour derived from `phyla_palette()`.

Usage

```
export_cytoscape(
  fancy_result,
  taxonomy,
  file_prefix = "fancy_network",
  edges = "thresholded"
)
```

Arguments

fancy_result	A FancyResult object as returned by fancy().
taxonomy	A data.frame with MAG IDs as row names and at least a Phyla column. Additional taxonomy ranks (Domain, Class, Order, Family, Genus, Species) are included when present.
file_prefix	Character; path prefix for the output files (default "fancy_network"). Two files are written: {prefix}_edges.tsv and {prefix}_nodes.tsv.
edges	Character; which edge set to export. "thresholded" (default) uses fancy_result$edges; "all" uses fancy_result$all_edges.

Value

Invisibly, a list with elements $nodes and $edges (data.frames matching the written files).

Examples

```
mock <- methods::new("FancyResult",
    edges = data.frame(
        source = c("MAG1", "MAG2"), target = c("MAG2", "MAG3"),
        HybridScore = c(0.9, 0.7)
    )
)
tax <- data.frame(
    Phyla = c("Bacillota", "Pseudomonadota", "Bacteroidota"),
    Genus = c("Genus_A", "Genus_B", "Genus_C"),
    row.names = c("MAG1", "MAG2", "MAG3")
)
out <- export_cytoscape(mock, tax,
    file_prefix = file.path(tempdir(), "ex")
)
head(out$edges)
```

---

fancy *Run the full Fancy hybrid network inference pipeline*

Description

Convenience wrapper that chains preprocessing, bootstrap network inference, edge scoring, and thresholding into a single call. Uses MI (via MRNET) and distance correlation with bootstrap resampling to produce a ranked, thresholded edge list.

Usage

```
fancy(
  data,
  n_bootstrap = 100L,
  k = 5L,
  find_k = FALSE,
  k_values = c(2L, 3L, 4L, 5L, 6L, 7L, 8L, 10L, 15L, 18L, 20L),
  cpus = 1L,
  use_mrnet = TRUE,
  w1 = 0.3,
  w2 = 0.7,
  score_type = "multiplicative",
  dcor_rescue_percentile = NULL,
  threshold_method = "quantile",
  threshold_value = 0.7,
  save_path = NULL,
  verbose = TRUE
)
```

Arguments

`data`	Numeric matrix or data.frame of CLR-transformed abundances with **samples as rows and MAGs as columns**. Pass `t(clr_normalize(counts))` or use the Samples x MAGs orientation directly.
`n_bootstrap`	Integer; number of bootstrap iterations (default 100).
`k`	Integer; kNN parameter for MI estimation (default 5).
`find_k`	Logical; if `TRUE`, run `find_optimal_k()` to choose k automatically via the elbow method (default `FALSE`).
`k_values`	Integer vector of k values to evaluate when `find_k = TRUE`.
`cpus`	Integer; number of parallel workers (default 1). Increase to parallelise bootstrap iterations across cores.
`use_mrnet`	Logical; if `TRUE` (default), apply MRNET to the MI matrix for sparse edge selection.
`w1`	Numeric weight for edge frequency in the hybrid score (default 0.3).
`w2`	Numeric weight for dCor stability in the hybrid score (default 0.7).
`score_type`	Character; `"multiplicative"` (default) or `"additive"`. See `hybrid_score()` for details.
`dcor_rescue_percentile`	Numeric between 0 and 1, or `NULL` (default). When set, edges with dCor stability above this percentile receive a minimum EdgeFrequency floor in multiplicative mode. See `hybrid_score()`.
`threshold_method`	Character; one of `"quantile"`, `"score"`, or `"top_n"` (default `"quantile"`).
`threshold_value`	Numeric; meaning depends on `threshold_method` (default 0.7, i.e. top 30 percent of edges for quantile method).

save_path	Optional directory path for saving bootstrap results.
verbose	Logical; if TRUE (default), emit progress messages.

Value

A FancyResult S4 object with slots:

edges The final thresholded edge data.frame.

all_edges All scored edges before thresholding (for re-thresholding).

all_edges_unfiltered All 4950 scored pairs before the hard frequency/stability filter. Useful for inspecting edges that were dropped by the pre-filter.

k The k value used.

n_bootstrap Number of bootstrap iterations run.

params List of all parameters for reproducibility.

Slots can be read with $ (e.g. result$edges) or @.

See Also

FancyResult

Examples

```
data(fancy_tiny_clr)
result <- fancy(t(fancy_tiny_clr), n_bootstrap = 2, cpus = 1)
head(result$edges)
plot(result)
```

---

fancy_tiny_clr *Tiny example CLR-transformed abundance table*

Description

Pre-computed centred log-ratio (CLR) transformed abundance table for 100 MAGs across 321 rumen samples. The CLR was computed on the full 2178-MAG compositional reference before subsetting, preserving the proper geometric mean. This is the primary input for network inference with fancy().

Usage

```
fancy_tiny_clr
```

Format

A data.frame with 100 rows (MAGs) and 321 columns (samples). Values are CLR-transformed (continuous, centred around zero).

Details

MAGs 1–4 are known nonlinear interaction pairs: MAGs 1–2 (Bulleidia vs AC2028) show threshold competitive exclusion, and MAGs 3–4 (RUG023 vs Cryptobacteroides) show an L-shaped nonlinear relationship.

Source

Subset from a published rumen metagenome study via `data-raw/make_datasets.R` with `set.seed(42)`.

Examples

```
data(fancy_tiny_clr)
dim(fancy_tiny_clr)
```

`fancy_tiny_counts` *Tiny example count table*

Description

Raw read-count table for the same 100 MAGs and 321 samples as fancy_tiny_clr. Provided for coverage filtering and exploratory analysis; network inference should use the pre-CLR matrix.

Usage

```
fancy_tiny_counts
```

Format

A data.frame with 100 rows (MAGs) and 321 columns (samples). Values are non-negative integers.

Source

Subset from a published rumen metagenome study via `data-raw/make_datasets.R` with `set.seed(42)`.

Examples

```
data(fancy_tiny_counts)
dim(fancy_tiny_counts)
```

fancy_tiny_coverage *Tiny example covered-fraction table*

Description

Genome covered-fraction table matching fancy_tiny_clr. Values represent the fraction of each MAG's reference genome covered by mapped reads in each sample.

Usage

```
fancy_tiny_coverage
```

Format

A data.frame with 100 rows (MAGs) and 321 columns (samples). Values are numeric fractions in [0, 1].

Source

Subset from a published rumen metagenome study via `data-raw/make_datasets.R` with `set.seed(42)`.

Examples

```
data(fancy_tiny_coverage)
summary(unlist(fancy_tiny_coverage))
```

fancy_tiny_metadata *Tiny example sample metadata*

Description

Per-sample metadata for the 321 samples in fancy_tiny_clr.

Usage

```
fancy_tiny_metadata
```

Format

A data.frame with 321 rows and 3 columns:

Sample Sample identifier (anonymised).

Breed Cattle breed (Aberdeen_Angus_X, Charolais_X, Limousin_X, or Luing).

CH4 Methane emissions in grams per day (continuous, range approximately 8–36).

Source

Subset from a published rumen metagenome study via `data-raw/make_datasets.R` with `set.seed(42)`.

Examples

```
data(fancy_tiny_metadata)
hist(fancy_tiny_metadata$CH4)
```

`fancy_tiny_taxonomy` *Tiny example GTDB taxonomy table*

Description

GTDB taxonomy for the 100 MAGs in fancy_tiny_clr. MAGs 1–2 are a nonlinear threshold-exclusion pair (Bulleidia vs AC2028, both Bacillota), and MAGs 3–4 are a nonlinear L-shaped pair (RUG023 / Spirochaetota vs Cryptobacteroides / Bacteroidota). The 100 MAGs span 9 phyla.

Usage

```
fancy_tiny_taxonomy
```

Format

A data.frame with 100 rows (MAGs as row names) and 7 columns: Domain, Phyla, Class, Order, Family, Genus, Species.

Source

Subset from a published rumen metagenome study via `data-raw/make_datasets.R` with `set.seed(42)`.

Examples

```
data(fancy_tiny_taxonomy)
table(fancy_tiny_taxonomy$Phyla)
```

FancyResult-class *FancyResult: S4 container for Fancy pipeline output*

Description

A formal S4 class holding the results of the `fancy()` hybrid network inference pipeline. Use the `fancy()` wrapper to construct objects of this class; slots can be read with the `$` operator (e.g. `result$edges`) or with standard slot access (`result@edges`).

Value

An object of class `FancyResult`.

Slots

`edges` A data.frame of thresholded edges (the final network).

`all_edges` A data.frame of all scored edges before thresholding (useful for re-thresholding).

`all_edges_unfiltered` A data.frame of all scored pairs before the hard frequency/stability pre-filter.

`k` Numeric; the kNN parameter used for MI estimation.

`n_bootstrap` Integer; number of bootstrap iterations run.

`params` A list of all pipeline parameters, for reproducibility.

See Also

`fancy()`

filter_mags *Filter MAGs by coverage and prevalence*

Description

Applies two filtering steps from the preprocessing pipeline:

1. **Coverage masking:** Binarises the coverage table at `min_coverage` and multiplies element-wise with the count table, zeroing out counts for MAG-sample pairs with insufficient genome coverage.
2. **Prevalence filter:** Retains only MAGs present (non-zero after masking) in at least `min_samples` samples.

Usage

```
filter_mags(count_table, coverage_table, min_coverage = 0.3, min_samples = 50L)
```

Arguments

count_table	A data.frame of read counts with MAGs as rows and samples as columns.
coverage_table	A data.frame of covered-fraction values with the same dimensions and names as count_table.
min_coverage	Numeric; minimum covered fraction to retain a count.
min_samples	Integer; minimum number of samples in which a MAG must be present after coverage masking.

Value

A data.frame of filtered counts (MAGs as rows, samples as columns).

Note

The original 02_Preprocessing.Rmd referenced count_table_cov on line 111, but the variable was actually count_table_cov_stats (line 105). This function eliminates that confusion.

Examples

```
data(fancy_tiny_counts)
data(fancy_tiny_coverage)
filt <- filter_mags(fancy_tiny_counts, fancy_tiny_coverage)
dim(filt)
```

---

find_optimal_k *Find optimal k for the kNN MI estimator*

Description

Tests multiple k values by computing the full MI matrix for each k and recording the average MI across all variable pairs. The resulting elbow curve helps choose k: the point where MI stops improving steeply (largest negative second derivative) is selected automatically when auto_select = TRUE.

Usage

```
find_optimal_k(
  data,
  k_values = c(2L, 3L, 4L, 5L, 6L, 7L, 8L, 10L, 15L, 18L, 20L),
  cpus = 1L,
  auto_select = TRUE
)
```

Arguments

data	A numeric matrix or data.frame with observations (samples) as rows and variables (MAGs) as columns.
k_values	Integer vector of k values to evaluate.
cpus	Integer; number of parallel workers.
auto_select	Logical; if TRUE (default), returns the elbow point as $best_k.

Details

Computation is parallelised across k values using **snowfall**.

Value

A list with:

results A data.frame(k, avg_mi) for plotting.

best_k (when auto_select = TRUE) the k at the elbow point, determined by the most negative second derivative.

Examples

```
small_mat <- matrix(rnorm(50), nrow = 10, ncol = 5)
res <- find_optimal_k(small_mat, k_values = c(2L, 3L), cpus = 1)
res$best_k
```

---

hybrid_score *Compute hybrid edge score*

Description

Scores edges by combining MRNET edge frequency with dCor stability. Two scoring modes are available:

"multiplicative" **(default)**

$$HybridScore = EdgeFrequency^{w1} \times Stability.dcor.scaled^{w2}$$

Edges need support from both components; a near-zero value in either component pulls the score toward zero.

"additive"

$$HybridScore = w1 \times EdgeFrequency + w2 \times Stability.dcor.scaled$$

Allows edges with strong dCor but weak MRNET frequency (or vice versa) to retain a meaningful score.

Usage

```
hybrid_score(
  edge_metrics,
  w1 = 0.3,
  w2 = 0.7,
  score_type = "multiplicative",
  dcor_rescue_percentile = NULL
)
```

Arguments

edge_metrics	A data.frame containing at least EdgeFrequency and Stability.dcor.scaled columns (as returned by compute_hybrid_scores()).
w1	Numeric weight for edge frequency (default 0.3).
w2	Numeric weight for dCor stability (default 0.7).
score_type	Character; "multiplicative" (default) or "additive".
dcor_rescue_percentile	Numeric between 0 and 1, or NULL (default). When set, edges with Stability.dcor.scaled above this percentile receive a minimum EdgeFrequency floor. Only applies to multiplicative scoring.

Details

When dcor_rescue_percentile is set (multiplicative mode only), edges whose Stability.dcor.scaled exceeds the given percentile have their EdgeFrequency raised to a minimum floor (the 30th percentile of non-zero EdgeFrequency values). This prevents near-zero MRNET frequency from suppressing edges that dCor finds reliably.

Value

The input data.frame with an added HybridScore column.

Examples

```
edges <- data.frame(
    source = c("MAG1", "MAG2", "MAG3"),
    target = c("MAG2", "MAG3", "MAG4"),
    EdgeFrequency = c(0.8, 0.5, 0.9),
    Stability.dcor.scaled = c(0.7, 0.3, 0.85)
)
scored <- hybrid_score(edges, w1 = 0.3, w2 = 0.7)
scored$HybridScore
```

`parse_count_tables` *Parse per-sample TSV files into combined MAG-by-sample matrices*

Description

Reads a directory of per-sample TSV files (as produced by read-mapping pipelines) and merges them into MAG-by-sample matrices for read counts, relative abundance, and covered fraction.

Usage

```
parse_count_tables(
  path,
  pattern = "*.tsv",
  columns = c(count = 1L, relative_abundance = 2L, covered_fraction = 3L),
  clean_names = TRUE,
  drop_first_row = TRUE
)
```

Arguments

`path`	Directory containing the TSV files.
`pattern`	Glob pattern passed to `base::list.files()`.
`columns`	Named integer vector mapping result names to column indices. Defaults to `c(count = 1L, relative_abundance = 2L, covered_fraction = 3L)`. Set to `NULL` or a subset to read only the matrices you need.
`clean_names`	Logical; apply `clean_sample_names()` to column names?
`drop_first_row`	Logical; drop the first data row (header duplication artifact in raw TSVs)?

Details

Each TSV is expected to have MAG identifiers as row names and at least three data columns (count, relative abundance, covered fraction). The first row in the raw files is often a duplicated header and is dropped by default (`drop_first_row = TRUE`).

Value

A named list of data.frames, one per entry in `columns`. Each data.frame has MAG IDs as row names and samples as columns.

Examples

```
d <- tempfile("tsvdir")
dir.create(d)
writeLines(
    c(
        "MAG\tcount\trel_abund\tcov_frac",
        "MAG1\t100\t0.5\t0.8", "MAG2\t200\t0.3\t0.6"
```

```
    ),
    file.path(d, "sample1.tsv")
)
tables <- parse_count_tables(d, drop_first_row = FALSE)
tables$count
```

`parse_gtdb_taxonomy` *Parse a GTDB-format taxonomy file*

Description

Reads a GTDB taxonomy TSV and cleans every rank by stripping rank prefixes (`d__`, `p__`, etc.) and removing sub-designation suffixes on Phyla and Genus (e.g. `Bacillota_C` becomes `Bacillota`). Species are simplified by removing the genus portion before the space and stripping trailing digits after `"sp"`.

Usage

```
parse_gtdb_taxonomy(file, strain_col = "Strain", fill_missing = TRUE)
```

Arguments

`file`	Path to a GTDB taxonomy TSV file.
`strain_col`	Character; name of the column containing MAG / strain identifiers (used as row names). Default `"Strain"`.
`fill_missing`	Logical; fill missing taxonomy levels as described above? Default `TRUE`.

Details

When `fill_missing = TRUE` (the default):

- Family `"NA"` is replaced with the Order value.
- Genus `"NA"` or `""` is replaced with the Family value.
- Species `""` is replaced with `"sp"`.

Value

A data.frame with `strain_col` values as row names and columns: Domain, Phyla, Class, Order, Family, Genus, Species.

Examples

```
tsv <- tempfile(fileext = ".tsv")
header <- paste(
    "Strain", "Domain", "Phyla", "Class", "Order",
    "Family", "Genus", "Species",
    sep = "\t"
)
row1 <- paste(
    "MAG1", "d__Bacteria", "p__Bacillota_C", "c__Clostridia",
    "o__Lachnospirales", "f__Lachnospiraceae", "g__Butyrivibrio",
    "s__Butyrivibrio sp1",
    sep = "\t"
)
writeLines(c(header, row1), tsv)
tax <- parse_gtdb_taxonomy(tsv)
tax$Phyla
```

---

phyla_palette *Assign colours to phyla*

Description

Returns a named character vector mapping each unique phylum to a colourblind-friendly hex colour. Colours are assigned in alphabetical order of phylum names so the mapping is deterministic.

Usage

```
phyla_palette(phyla)
```

Arguments

phyla	Character vector of phylum names (duplicates are OK).

Value

A named character vector of hex colours, one per unique phylum (sorted alphabetically).

Examples

```
phyla_palette(c("Bacillota", "Pseudomonadota", "Bacillota"))
```

`plot,FancyResult,missing-method`
Plot the hybrid score distribution of a FancyResult

Description

Plot method for FancyResult objects returned by `fancy()`. Draws a histogram of `HybridScore` across all scored edges with a vertical line at the threshold cutoff. The number of retained edges is annotated.

Usage

```
## S4 method for signature 'FancyResult,missing'
plot(x, y, ...)
```

Arguments

`x`	A FancyResult object as returned by `fancy()`.
`y`	Ignored; present for compatibility with the `plot()` generic.
`...`	Additional arguments passed to `graphics::hist()`.

Value

Invisibly returns `x`.

Examples

```
mock <- methods::new("FancyResult",
    edges = data.frame(source = "A", target = "B", HybridScore = 0.9),
    all_edges = data.frame(
        source = paste0("M", seq_len(20)),
        target = paste0("M", 21:40),
        HybridScore = runif(20)
    ),
    params = list(threshold_method = "quantile", threshold_value = 0.7)
)
plot(mock)
```

plot_k_elbow *Plot elbow curve for k selection*

Description

Takes the list returned by `find_optimal_k()` and plots the elbow curve (k vs average MI). When `$best_k` is present, marks the elbow point with a distinct symbol and label.

Usage

```
plot_k_elbow(k_results)
```

Arguments

`k_results`	A list as returned by `find_optimal_k()`, containing `$results` (data.frame with `k` and `avg_mi` columns) and optionally `$best_k`.

Value

Invisibly returns `k_results`.

Examples

```
k_res <- list(
    results = data.frame(k = c(3, 5, 7, 10), avg_mi = c(0.5, 0.45, 0.42, 0.41)),
    best_k = 5
)
plot_k_elbow(k_res)
```

plot_network *Plot the co-abundance network*

Description

Draws a network graph from a `"fancy"` result using igraph. Nodes are coloured by phylum (via `phyla_palette()`) and labelled with genus names. Edge width is proportional to `HybridScore`. When `community = TRUE`, modularity-based community detection is applied and clusters are shown as shaded convex hulls around groups of co-abundant MAGs.

Usage

```
plot_network(
  fancy_result,
  taxonomy,
  top_n = NULL,
  layout = "fruchterman.reingold",
  community = FALSE
)
```

Arguments

fancy_result	A FancyResult object as returned by fancy().
taxonomy	A data.frame with MAG IDs as row names and at least Phyla and Genus columns.
top_n	Integer or NULL; if set, only the top N edges by HybridScore are plotted (default NULL = all thresholded edges).
layout	Character; igraph layout algorithm name (default "fruchterman.reingold").
community	Logical; if TRUE, run modularity-based community detection and draw shaded convex hulls around clusters (default FALSE).

Value

Invisibly returns the igraph graph object.

Examples

```
data(fancy_tiny_clr)
data(fancy_tiny_taxonomy)
result <- fancy(t(fancy_tiny_clr), n_bootstrap = 2, cpus = 1)
plot_network(result, fancy_tiny_taxonomy)
```

---

show,FancyResult-method

Display a FancyResult

Description

Compact summary method for FancyResult objects.

Usage

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

Arguments

object	A FancyResult object.

Value

`object`, invisibly.

`threshold_edges` *Threshold scored edges*

Description

Filters a scored edge list using one of three methods:

`"quantile"` Retain edges above the given quantile of `HybridScore`. Default `value = 0.7` retains the top 30 percent.

`"score"` Retain edges with `HybridScore >= value`.

`"top_n"` Retain the top `value` edges by `HybridScore`.

Usage

```
threshold_edges(scored_edges, method = "quantile", value = 0.7)
```

Arguments

`scored_edges`	A data.frame with a `HybridScore` column.
`method`	Character; one of `"quantile"`, `"score"`, `"top_n"`.
`value`	Numeric; meaning depends on `method`.

Value

A data.frame of retained edges, sorted by descending `HybridScore`.

Examples

```
scored <- data.frame(
    source = paste0("MAG", 1:10),
    target = paste0("MAG", 11:20),
    HybridScore = seq(0.1, 1.0, by = 0.1)
)
top5 <- threshold_edges(scored, method = "top_n", value = 5)
nrow(top5)
```

Index

∗ **datasets**
    fancy_tiny_clr, 8
    fancy_tiny_counts, 9
    fancy_tiny_coverage, 10
    fancy_tiny_metadata, 10
    fancy_tiny_taxonomy, 11
$,FancyResult-method, 2

base::list.files(), *16*
bootstrap_networks, 3

clean_sample_names, 4
clean_sample_names(), *16*
clr_normalize, 5
compositions::clr(), *5*
compute_hybrid_scores(), *15*
compute_mi_matrix(), *3*

export_cytoscape, 5

fancy, 6
fancy(), *6*, *8*, *12*, *19*, *21*
fancy_tiny_clr, 8, *9–11*
fancy_tiny_counts, 9
fancy_tiny_coverage, 10
fancy_tiny_metadata, 10
fancy_tiny_taxonomy, 11
FancyResult, *2, 3*, *6*, *8*, *19*, *21*
FancyResult-class, 12
filter_mags, 12
find_optimal_k, 13
find_optimal_k(), *7*, *20*

graphics::hist(), *19*

hybrid_score, 14
hybrid_score(), *7*

parse_count_tables, 16
parse_gtdb_taxonomy, 17
phyla_palette, 18
phyla_palette(), *5*, *20*
plot(), *19*
plot,FancyResult,missing-method, 19
plot_k_elbow, 20
plot_network, 20

show,FancyResult-method, 21

threshold_edges, 22

utils::read.csv(), *4*