

Package: levi (via r-universe)

September 20, 2026

Type Package

Title Landscape Expression Visualization and Network Inference

Version 1.99.0

Description Projects gene expression onto the plane of a biological network and draws it as a landscape: a Gaussian-kernel weighted surface on an absolute scale where 0.5 means no change. Regions of the landscape are tested by permutation, either conditional on the network and layout (node-label shuffling, maximum regional mass) or against biological replication (sample-label permutation, with blocking). Graph-native tests complement the landscape: threshold-free cluster enhancement on network components, Moran and Getis-Ord autocorrelation, Laplacian smoothness, Freedman-Lane residual permutation with covariates and degree-preserving rewiring nulls. Pseudobulk workflows for single-cell RNA-seq permute donors rather than cells. Adapters read DESeq2, edgeR, limma and Seurat results, SummarizedExperiment objects and STRING networks. A Shiny interface covers the interactive visualization.

URL <https://bioconductor.org/packages/levi>

BugReports <https://github.com/jrybarczyk/levi/issues>

Encoding UTF-8

LazyData false

Imports BiocParallel, Matrix, dplyr (>= 0.7.4), ggplot2 (>= 2.2.1), grid, grDevices, igraph (>= 1.2.1), methods, Rcpp (>= 0.12.18), reshape2 (>= 1.4.3), shiny (>= 1.0.5), stats, SummarizedExperiment, utils, withr, xml2 (>= 1.2.0)

Suggests testthat, BiocStyle, knitr, rmarkdown, DT (>= 0.4), shinydashboard (>= 0.7.0), shinyjs (>= 1.0), plotly (>= 4.9.0), ggrepel, patchwork, cowplot, gridExtra, htmlwidgets, magick, reticulate, Biobase, DESeq2, edgeR, limma, Seurat, SingleCellExperiment, S4Vectors, clusterProfiler, org.Hs.eg.db, STRINGdb

LinkingTo Rcpp

License GPL (>= 2)

biocViews GeneExpression, Microarray, Sequencing, RNASeq, Network,
GraphAndNetwork, StatisticalMethod, Software, Visualization,
GUI, GeneSetEnrichment, DifferentialExpression, SingleCell,
Proteomics

VignetteBuilder knitr

LazyLoad yes

NeedsCompilation yes

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev zlib1g-dev

Repository <https://bioc.r-universe.dev>

Date/Publication 2026-09-19 22:48:21 UTC

RemoteUrl <https://github.com/bioc/levi>

RemoteRef HEAD

RemoteSha ac8c924642aac186de9e2ca9ccc0b8652fd5ab5c

Contents

bimodal_dataset	3
flat_dataset	4
gradient_dataset	5
hub_dataset	7
levi	8
leviAdjustPathways	13
leviBulkGraphInference	14
leviDiff	16
leviEnrich	18
leviFromBioc	19
leviFromDESeq2	20
leviFromEdgeR	21
leviFromEdges	22
leviFromExpressionSet	24
leviFromLimma	25
leviFromSE	26
leviFromSeurat	28
leviFromSTRING	29
leviGraphClusterInference	31
leviGraphGetisOrd	33
leviGraphMoran	34
leviGraphRewiringInference	35
leviGraphSpectrum	36
leviGraphTFCEFreemanLane	37
leviGraphTFCEInference	38
leviGraphTFCERewiring	40

leviGraphWeightedTopology	41
leviGrid	42
leviPseudobulk	43
leviRegionGenes	44
leviReplicateInference	45
leviSave3D	46
leviSingleCellGraphInference	48
leviSingleCellInteractionTFCE	49
leviSingleCellRegionalInference	51
leviSingleCellTFCEInference	53
leviSingleCellTopologyInference	55
LEVlui	56
logfc_dataset	57
multicomp_dataset	59
print.levi_result	60
readExpColumn	61
sparse_dataset	62

Index 64

bimodal_dataset	<i>Bimodal toy dataset</i>
-----------------	----------------------------

Description

A 13-node network with two star clusters connected by a neutral bridge gene. Cluster A is strongly over-expressed; cluster B is strongly under-expressed. The resulting landscape has a peak on the left and a valley on the right - ideal for validating simultaneous peak-and-valley detection and [leviDiff](#).

Format

Two tab-delimited files in inst/extdata/:

bimodal_network.dat Two star sub-networks: A_HUB + A1-A5 (left, $x \sim 0.1-0.35$) and B_HUB + B1-B5 (right, $x \sim 0.65-0.90$), linked via BRIDGE ($x = 0.50$).

bimodal_expression.dat Columns ID, Test, Control. Cluster A: Test = 200, Control = 10. BRIDGE: Test = Control = 100 (neutral). Cluster B: Test = 10, Control = 200.

Value

NULL (this object documents data files, not a function)

Expected results

- `score(A_HUB) > score(BRIDGE) > score(B_HUB)`.
- All cluster-A nodes score > 0.5 ; all cluster-B nodes < 0.5 .
- Peak(s) detected near A cluster; valley(s) near B cluster.
- With `perm_side = "both"`: dashed contour on left, dotted contour on right.

Source

Simulated data generated for unit testing.

See Also

[levi](#), [leviDiff](#)

Examples

```
bim_net <- file.path(system.file(package = "levi"),
                     "extdata", "bimodal_network.dat")
bim_expr <- file.path(system.file(package = "levi"),
                      "extdata", "bimodal_expression.dat")

res <- levi(
  networkCoordinatesInput = bim_net,
  expressionInput         = bim_expr,
  fileTypeInput           = "dat",
  geneSymbolInput         = "ID",
  readExpColumn           = readExpColumn("Test-Control"),
  contrastValueInput      = 50,
  resolutionValueInput    = 50,
  zoomValueInput          = 50,
  smoothValueInput        = 50,
  contourLevi             = TRUE
)
```

flat_dataset	<i>Flat (null-expression) toy dataset</i>
--------------	---

Description

A 3x3 grid network where all genes have identical expression (Test = Control = 100). The landscape is expected to be nearly flat (all scores ~ 0.5) with no meaningful peaks or valleys. Used to verify that `levi` does not produce false positives when there is no differential expression.

Format

Two tab-delimited files in `inst/extdata/`:

`flat_network.dat` Nine nodes F11-F33 arranged as a 3x3 grid with horizontal and vertical edges (12 edges total). Node positions span (0.20, 0.20) to (0.80, 0.80).

`flat_expression.dat` All 9 genes with Test = Control = 100, yielding a uniform signal ratio of 0.5.

Value

NULL (this object documents data files, not a function)

Expected results

- All landscape scores within [0.2, 0.8] (no extreme regions).
- Standard deviation of scores < 0.15.
- No peaks with score > 0.8; no valleys with score < 0.2.

Source

Simulated data generated for unit testing.

See Also

[levi](#)

Examples

```
flat_net <- file.path(system.file(package = "levi"),
                      "extdata", "flat_network.dat")
flat_expr <- file.path(system.file(package = "levi"),
                      "extdata", "flat_expression.dat")

res <- levi(
  networkCoordinatesInput = flat_net,
  expressionInput         = flat_expr,
  fileTypeInput           = "dat",
  geneSymbolInput         = "ID",
  readExpColumn           = readExpColumn("Test-Control"),
  contrastValueInput      = 50,
  resolutionValueInput    = 50,
  zoomValueInput          = 50,
  smoothValueInput        = 50
)
round(res$scores$LandscapeScore, 3)
```

gradient_dataset

Gradient-topology toy dataset

Description

A 6-node linear chain where expression increases monotonically from left (GA) to right (GF). Designed to verify that landscape scores preserve strict rank order along a gradient and that no spurious peaks appear in intermediate positions.

Format

Two tab-delimited files in inst/extdata/:

gradient_network.dat Linear chain GA-GB-GC-GD-GE-GF with nodes equally spaced from $x = 0.10$ to $x = 0.90$ at $y = 0.50$.

gradient_expression.dat Columns ID, Test, Control. Values increase from GA (Test = 5, Control = 100) to GF (Test = 100, Control = 5), giving a smooth gradient of signal ratios from ~ 0.09 to ~ 0.91 .

Value

NULL (this object documents data files, not a function)

Expected results

- Strictly monotone scores: $GA < GB < GC < GD < GE < GF$.
- GF ranked 1st; GA ranked last.
- $score(GF) > 0.5$; $score(GA) < 0.5$.
- No peak detected at intermediate nodes.

Source

Simulated data generated for unit testing.

See Also

[levi](#), [hub dataset](#)

Examples

```
grad_net <- file.path(system.file(package = "levi"),
                      "extdata", "gradient_network.dat")
grad_expr <- file.path(system.file(package = "levi"),
                      "extdata", "gradient_expression.dat")

res <- levi(
  networkCoordinatesInput = grad_net,
  expressionInput         = grad_expr,
  fileTypeInput           = "dat",
  geneSymbolInput         = "ID",
  readExpColumn           = readExpColumn("Test-Control"),
  contrastValueInput      = 50,
  resolutionValueInput    = 50,
  zoomValueInput          = 50,
  smoothValueInput        = 50
)
res$scores[order(res$scores$Rank), c("Gene", "LandscapeScore", "Rank")]
```

hub_dataset	<i>Hub-topology toy dataset</i>
-------------	---------------------------------

Description

A minimal 9-node star network with a known, asymmetric expression pattern. The central hub and its four closest spokes are strongly over-expressed (Test = 200, Control = 10); the four outer corner nodes are strongly under-expressed (Test = 5, Control = 200). The expected landscape shows a clear peak at the centre and valleys at the periphery.

Format

Two tab-delimited files in `inst/extdata/`:

`hub_network.dat` Medusa DAT network file with 9 nodes (HUB, N1-N8) and 8 edges (all connecting to HUB). Node coordinates place the hub at (0.50, 0.50) and spokes at increasing radial distances.

`hub_expression.dat` Tab-delimited expression file with columns ID, Test, Control. Inner nodes (HUB, N1-N4): Test = 200, Control = 10. Outer nodes (N5-N8): Test = 5, Control = 200.

Value

NULL (this object documents data files, not a function)

Expected results

With default parameters (`signal_mode = "ratio"`):

- `score(HUB) > all spoke scores > 0.5`
- `score(N5) ~ score(N6) ~ score(N7) ~ score(N8) < 0.5`
- Peaks detected over the over-expressed core (HUB and N1-N4 all reach the same maximum, so several are reported); valleys near N5-N8.

Source

Simulated data generated for unit testing. No biological meaning.

See Also

[levi](#), [gradient dataset](#)

Examples

```

hub_net <- file.path(system.file(package = "levi"),
                     "extdata", "hub_network.dat")
hub_expr <- file.path(system.file(package = "levi"),
                     "extdata", "hub_expression.dat")

res <- levi(
  networkCoordinatesInput = hub_net,
  expressionInput         = hub_expr,
  fileTypeInput           = "dat",
  geneSymbolInput         = "ID",
  readExpColumn           = readExpColumn("Test-Control"),
  contrastValueInput      = 50,
  resolutionValueInput    = 50,
  zoomValueInput          = 50,
  smoothValueInput        = 50
)
head(res$scores)

```

levi

levi

Description

This is the Levi script mode. It allows you to create the integration of networks and gene expression levels as batch processing

Usage

```

levi(
  expressionInput,
  fileTypeInput,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  geneSymbolInput,
  readExpColumn,
  contrastValueInput = 50,
  zoomValueInput = 50,
  resolutionValueInput = 50,
  smoothValueInput = 50,
  expressionLog = FALSE,
  contourLevi = FALSE,
  setcolor = "default",
  plot3d = FALSE,
  n_perm = 0,
  sig_level = 0.05,

```

```

perm_side = c("both", "over", "under"),
signal_mode = c("ratio", "logfc", "zscore"),
logfc_k = 1,
p_adjust_method = "BY",
region_threshold = 0.1,
region_min_cells = 3L,
inference_unit = c("region", "cell"),
perm_strata = NULL,
edge_weighting = c("midpoint", "degree", "none"),
.parsed_network = NULL,
.draw = TRUE
)

```

Arguments

- expressionInput** File path to gene expression data (tab-delimited), OR a `data.frame` / `matrix` directly. When passing a `data.frame`, it must contain a gene identifier column and at least one expression column. Objects from `leviFromDESeq2()`, `leviFromEdgeR()`, `leviFromLimma()`, and `leviFromSeurat()` are accepted directly.
- fileTypeInput** Format of the biological network, given as one of "dat" (Medusa), "dyn" (RedeR), "net" (Pajek) or "stg" (STRING/STITCH). This is the format code, not a file name: the network itself is passed in `networkCoordinatesInput`.
- networkCoordinatesInput** File path to the biological network carrying the node coordinates, OR a `data.frame` with them directly (as returned in `$nodes` by `leviFromSTRING()`). The format must match `fileTypeInput`.
- networkInteractionsInput** Parameter available only to STRING/STITCH data format. It allows the user to load the interaction data file of the network.
- geneSymbolInput** Column name from gene expression data containing the identifier (gene Symbol, Entrez ID, EMSEMBL, etc).
- readExpColumn** Variable from `readExpColumn` function containing the comparisons of the experiments
- contrastValueInput** Numeric value controlling how tightly the landscape hugs the network. The landscape is built by normalised convolution, and this parameter sets the occupancy threshold below which a grid cell is considered background and returned as NA: higher values give a tighter silhouette. The variable range is 0 to 100. The default value is 50.
- zoomValueInput** Numeric value for image zoom, from 0 (widest frame) to 100 (tightest). The grid always leaves enough margin around the network for the smoothed silhouette to fit, whatever the smoothing and contrast; the zoom adds up to 20% of the network extent on top of that margin. The default value is 50.

resolutionValueInput	Numeric value for image resolution. The variable range is 0 to 100. The default value is 50.
smoothValueInput	Numeric value for image smoothness. Sets the width (σ) of the Gaussian kernel used by the normalised convolution, scaled relative to resolutionValueInput so that changing the resolution does not change the apparent smoothing. Unlike in levi < 2.0.0, this parameter no longer affects the value scale, only the level of detail. The variable range is 0 to 100. The default is 50.
expressionLog	Logical. In ratio mode, TRUE back-transforms log2 expression with 2^x before calculation. Ignored in logfc and zscore modes, which use log-scale columns directly. Default FALSE.
contourLevi	Logical variable to allow contour lines. The default is FALSE.
setcolor	Select the color palette to build the heatmap. There is two options the **Multicolor** has 20 color levels combined. The **Two colors** has two types of color and the options available are: <i>*purple_pink*</i> , <i>*green_blue*</i> , <i>*blue_yellow*</i> , <i>*pink_green*</i> , <i>*orange_purple*</i> , <i>*green_marine*</i> .
plot3d	Logical. When TRUE, an interactive 3D surface plot is additionally generated using plotly. Requires the plotly package. Default is FALSE.
n_perm	Integer. Number of node-label permutations, default 0. Measured node expression pairs are shuffled together; missing positions stay fixed and edge signals are recalculated. Tests spatial association conditional on the chosen network/layout, not differential expression between replicates. Raw Monte Carlo p-values are $(count + 1)/(n_perm + 1)$. A larger number improves resolution but does not establish biological validity.
sig_level	Numeric (0-1). Significance threshold for the contour overlay. A contour is drawn wherever the p-value equals sig_level. Default is 0.05.
perm_side	Character. Which direction to test when n_perm > 0. Options: <i>"both"</i> (default) dashed contour for over-expression + dotted for under-expression. <i>"over"</i> Only the dashed contour. <i>"under"</i> Only the dotted contour. result\$pvalues is a list with \$over and \$under matrices when inference_unit = "cell", or NULL when n_perm = 0 or in regional mode, where perm_side selects which significant regions are outlined.
signal_mode	Character. How raw expression values are converted to the landscape score in [0, 1]. Choose based on your data type: <i>"ratio"</i> (default) $Test/(Test + Control)$, without min-max normalisation; equal nonzero values map to 0.5. Best for linear-scale data: raw counts, TPM, FPKM, linear proteomics LFQ. Also correct when expressionLog = TRUE is used to back-transform log2 microarray intensities into linear scale. <i>"logfc"</i> Sigmoid transformation: $1/(1 + e^{-k \cdot \log FC})$ where $k = \log fc_k$ and $\log FC = Test - Control$ for two log-scale columns, or the supplied logFC in single-column mode. Score 0.5 = no change; > 0.5 = up-regulated; < 0.5 = down-regulated. Best for: RMA microarray, VST/rlog from DESeq2, log2 proteomics, scRNA-seq avg_log2FC (single-column mode with readExpColumn("logFC-logFC")).

	"zscore" Z-scores the logFC distribution then maps to $[0, 1]$ via <code>pnorm()</code>. Score 0.5 = mean logFC of measured nodes and derived edge support points; tails approach 0/1. This is relative within each comparison and uses mean/SD, not a robust estimator. It does not encode an absolute no-change baseline.
logfc_k	Numeric. Steepness of the sigmoid in <code>signal_mode = "logfc"</code>. Larger values make the transition from 0 to 1 sharper (more contrast between up/down). Default is 1. Typical range: 0.5-3. Use $k = 0.5$ for datasets with large logFC values (e.g., scRNA-seq fold-changes often range ± 5) and $k = 2$ for tight fold-changes (e.g., ± 1 in microarray).
p_adjust_method	Multiple-testing correction used only when <code>inference_unit = "cell"</code>, applied jointly to both directional families over occupied grid cells in each comparison. Default "BY" accommodates arbitrary dependence; "BH", "holm", "none" and other methods accepted by <code>stats::p.adjust</code> are available. Contours and pvalues use these adjusted values; <code>raw_pvalues</code> is unadjusted.
region_threshold	Positive deviation from the neutral score 0.5 used to define a regional cell. The default 0.1 identifies cells ≥ 0.6 or ≤ 0.4. This parameter describes regions and does not assign statistical significance.
region_min_cells	Minimum number of eight-connected grid cells retained as a region. Default is 3.
inference_unit	Either "region" (default) or "cell" (legacy). With "region" and <code>n_perm > 0</code>, regions are redetected in every node-label permutation and the observed mass of each region is compared with the maximum mass across all regions and both directions. Results are in <code>regions\$summary\$PSpatial</code> and <code>\$Significant</code>; <code>regions\$null_max_mass</code> stores the null maxima. This controls the search under the global exchangeable spatial null, conditional on network/layout. It does not establish significance of individual cells or genes, precise boundaries, or differences between biological replicates. <code>pvalues</code> and <code>raw_pvalues</code> are NULL in regional mode; the 2D plot labels areas rather than named gene peaks and white boundaries mark regions passing <code>sig_level</code>. The 3D plot shows the descriptive surface without regional boundaries. "cell" tests every occupied grid cell separately and adjusts the two directional families jointly with <code>p_adjust_method</code>. Cells are strongly correlated, so that adjustment is very conservative; the mode is kept for the graphical interface and for comparison with earlier versions. See <code>vignette("levi_inference")</code> for which null answers which question.
perm_strata	Optional vector defining exchangeability strata for the node-label permutation. A named vector is matched to network gene identifiers; an unnamed vector must follow network-node order. Signals are shuffled only among measured nodes in the same stratum. Useful for bins of mean expression, detection rate or network degree. Default 'NULL' permutes all measured nodes.
edge_weighting	Character. How the edge midpoints enter the landscape. "midpoint" (default, the historical behaviour) gives every node and every edge midpoint the same weight, so a hub of degree d surrounds itself with d support points and dominates its neighbourhood. "degree" weights the midpoint of edge (i, j) by

$(1/d_i + 1/d_j)/2$, so the midpoints around any node add up to one whatever its degree. "none" drops the midpoints and smooths the node values alone. All three keep the same coordinates, silhouette logic and permutation nulls; the choice is recorded in `metadata$edge_weighting` and reused by the sample-label tests.

`.parsed_network`, `.draw`

Internal controls used to reuse a parsed network and suppress repeated drawing in resampling workflows.

Details

Integrates the biological network and gene expression levels (or other type of data). In single-column ratio mode, the score is $\text{abundance}/(\text{abundance} + 1)$, with no control comparison. Missing measurements and undefined ratios are assigned 0.5. Surface, scores, peaks and tests use the same Gaussian-smoothed signal. A gene's smoothed score also reflects its neighbours, so a locally unchanged gene can receive a non-neutral score.

Every edge contributes a support point at its midpoint, carrying the mean of its two endpoints. With the default `edge_weighting = "midpoint"` a hub with many edges places many support points around itself and dominates the local average: the landscape is implicitly weighted by degree. The node-label permutation test keeps the network fixed, so this weighting is part of its null and does not bias the inference, but it does shape what the eye reads on the figure. `edge_weighting = "degree"` removes that emphasis while keeping the edges as carriers of neighbourhood signal; "none" smooths the nodes alone.

Value

Invisibly returns a list (or a list of such lists, one per comparison, when `readExpColumn` carries more than one) containing:

comparison Name of the comparison (e.g. "Tumor-Normal").

landscape `data.frame` with the plotted surface in long format (`Var1`, `Var2`, `z`). Cells outside the network silhouette are NA, not 0. Required by `leviDiff`.

scores `data.frame` with one row per network node (`Gene`, `X`, `Y`, `LandscapeScore`, `Rank`), ordered from the highest score to the lowest.

peaks `data.frame` with the automatically detected peaks and valleys (`Type`, `NearestGene`, `MatrixRow`, `MatrixCol`, `Score`).

pvalues list with the `$over` and `$under` permutation p-value matrices in cell mode, or NULL when `n_perm = 0` or `inference_unit = "region"`.

raw_pvalues Unadjusted directional permutation p-values, or NULL.

regions List with summary (one row per eight-connected area) and cells (member grid cells). In regional inference mode, the summary also contains `PSpatial` and `Significant`; `null_max_mass` and inference record the null and method. Otherwise regions are descriptive.

metadata Signal meaning, parameters, network, coordinates, missing genes, RNG state before calculation and software versions. Call `set.seed()` before `levi()` for reproducibility.

plot the ggplot object, also drawn on the active device.

plot3d the plotly surface when `plot3d = TRUE`, otherwise NULL. Like `plot`, it is also shown when produced. To save it from a chosen viewpoint, set the camera and write the file:

```
p <- plotly::layout(result$plot3d, scene = list(camera = list(
  eye = list(x = 1.6, y = -1.6, z = 0.9))))
htmlwidgets::saveWidget(p, "surface.html") # interactive, keeps the view
leviSave3D(p, "surface.png")                # static; needs kaleido
```

eye is the camera position relative to the centre of the surface; larger values move it away. Interactively, rotate the surface and use the camera icon of the plotly toolbar to save the current view as PNG.

Author(s)

Isabelle Mira da Silva (isabelle.silva@unesp.br), Jose Rafael Pilan (rafael.pilan@unesp.br)

Examples

```
template_network <- file.path(system.file(package="levi"), "extdata",
  "medusa.dat", fsep = .Platform$file.sep)

template_expression <- file.path(system.file(package="levi"),
  "extdata", "expression.dat", fsep = .Platform$file.sep)

multicolor <- levi(networkCoordinatesInput = template_network,
  expressionInput = template_expression, fileTypeInput = "dat",
  geneSymbolInput = "ID",
  readExpColumn=readExpColumn("TumorCurrentSmoker-NormalNeverSmoker"),
  contrastValueInput = 50, resolutionValueInput = 50, zoomValueInput = 50,
  smoothValueInput = 50, expressionLog = FALSE, contourLevi = TRUE)

twocolors <- levi(networkCoordinatesInput = template_network,
  expressionInput = template_expression, fileTypeInput = "dat",
  geneSymbolInput = "ID",
  readExpColumn = readExpColumn("TumorCurrentSmoker-NormalNeverSmoker"),
  setcolor = "pink_green", contourLevi = FALSE)
```

leviAdjustPathways	<i>Adjust regional LEVI results across pathways or networks</i>
--------------------	---

Description

Applies a second multiplicity correction to regional spatial P-values from several independently fitted LEVI networks. Each input result must use regional inference. The within-network P-values already use the maximum regional mass; this function controls the additional family of networks.

Usage

```
leviAdjustPathways(results, method = "holm", level = 0.05)
```

Arguments

results	Named list of 'levi_result' objects.
method	Method accepted by [stats::p.adjust()]. Default "holm".
level	Significance threshold.

Value

The input list with 'PFamily' and 'FamilySignificant' added to every regional summary.

Examples

```
set.seed(1)
net <- system.file("extdata", "hub_network.dat", package = "levi")
genes <- c("HUB", paste0("N", 1:8))
run <- function(lfc) levi(expressionInput = data.frame(ID = genes, logFC = lfc),
  networkCoordinatesInput = net, fileTypeInput = "dat", geneSymbolInput = "ID",
  readExpColumn = readExpColumn("logFC-logFC"), signal_mode = "logfc",
  inference_unit = "region", n_perm = 3, resolutionValueInput = 1,
  region_threshold = .02)
results <- list(pathway1 = run(rnorm(9, 1)), pathway2 = run(rnorm(9, -1)))
adjusted <- leviAdjustPathways(results)
adjusted$pathway1$regions$summary
```

leviBulkGraphInference

Bulk RNA-seq graph-cluster inference

Description

Fits a DESeq2 negative-binomial model to raw counts, projects the signed Wald statistic onto a biological graph, and assigns maximum-cluster-mass P-values by permuting sample labels. It is intended for a two-group design without additional covariates. When 'blocks' is supplied, the block is included as a fixed effect in the DESeq2 model and labels are permuted only within blocks. This supports paired two-condition studies.

Usage

```
leviBulkGraphInference(
  counts,
  groups,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  test = unique(groups)[2],
  control = unique(groups)[1],
  threshold = 2,
  n_perm = 999L,
```

```

    blocks = NULL,
    seed = NULL,
    permutation_method = c("auto", "monte_carlo", "exact"),
    max_exact = 50000L,
    refit_dispersions = FALSE,
    BPPARAM = SerialParam()
)

```

Arguments

counts	Integer genes-by-samples count matrix with gene row names.
groups	Two-level group vector.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	Network inputs.
test, control	Group labels defining test minus control.
threshold	Absolute Wald-statistic threshold for graph components.
n_perm	Number of sample-label permutations.
blocks	Optional exchangeability blocks.
seed	Optional RNG seed.
permutation_method	"auto", "monte_carlo" or "exact". "exact" enumerates every distinct label assignment; "monte_carlo" draws 'n_perm' random permutations; "auto" uses the exact scheme when the number of assignments is at most 'max_exact' and Monte Carlo otherwise.
max_exact	Maximum number of label assignments enumerated by the exact permutation scheme.
refit_dispersions	Logical. 'FALSE' (default) keeps the size factors and dispersions estimated on the observed labels and refits only the negative-binomial GLM in each permutation, a test conditional on those estimates and considerably faster. 'TRUE' re-estimates dispersions in every permutation.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Details

For a fast alternative on the same design, transform the counts with 'limma::voom()' or 'edgeR::cpm(log = TRUE)' and use [leviGraphClusterInference()], whose moderated t refits in milliseconds.

Value

List containing the DESeq2 statistic, graph regions, null maxima and the dispersion estimator used ("trend" for the standard 'DESeq()' pipeline, "genewise" when the trend could not be fitted).

Examples

```
if (requireNamespace("DESeq2", quietly = TRUE)) {

  set.seed(3)
  genes <- c("HUB", paste0("N", 1:8))
  counts <- matrix(rpois(length(genes) * 6, 30), length(genes), 6,
    dimnames = list(genes, paste0("s", 1:6)))
  counts[, 4:6] <- counts[, 4:6] + 30L
  out <- leviBulkGraphInference(counts, c("C", "C", "C", "T", "T", "T"),
    system.file("extdata", "hub_network.dat", package = "levi"),
    test = "T", control = "C", threshold = 1, n_perm = 3, seed = 1)
  out$regions$summary

}
```

leviDiff

leviDiff

Description

Compute a differential landscape by subtracting two `levi()` results cell by cell. The resulting map highlights regions where expression changed **between** conditions A and B.

Usage

```
leviDiff(
  result_a,
  result_b,
  label_a = NULL,
  label_b = NULL,
  setcolor = "default"
)
```

Arguments

<code>result_a</code>	A single <code>levi()</code> result (list with <code>\$landscape</code>).
<code>result_b</code>	A single <code>levi()</code> result to subtract from <code>result_a</code> .
<code>label_a</code>	Character. Label for condition A. Default: uses <code>result_a\$comparison</code> or "A".
<code>label_b</code>	Character. Label for condition B. Default: uses <code>result_b\$comparison</code> or "B".
<code>setcolor</code>	Character palette name accepted by <code>levi()</code> . Default "default" uses blue-to-red.

Details

Both results must have been generated with the **same** network and **same** resolution (resolutionValueInput). The Diff value is positive where condition B is more expressed and negative where condition A is more expressed.

Both results must contain compatible metadata: signal mode, input scale, network/layout, grid/smoothing, missing genes and versions. Legacy results without metadata must be recalculated. zscore differences compare separately standardised relative positions and issue a warning.

Value

Invisibly returns a list with:

`diff` data.frame with columns X, Y, Diff (B minus A, range -1 to 1)

`plot` ggplot2 differential landscape object

`comparison` character string summarising the subtraction

Examples

```
hub_n <- system.file("extdata", "hub_network.dat", package = "levi")
hub_e <- system.file("extdata", "hub_expression.dat", package = "levi")
res <- levi(expressionInput      = hub_e,
             networkCoordinatesInput = hub_n,
             fileTypeInput       = "dat",
             geneSymbolInput     = "ID",
             readExpColumn       = readExpColumn("Test-Control"),
             resolutionValueInput = 10,
             smoothValueInput    = 5)
diff_res <- leviDiff(res, res)
head(diff_res$diff)

# Two comparisons from the same network, then their differential landscape
multi_e <- system.file("extdata", "hub_multicomp_expression.dat",
                       package = "levi")
res2 <- levi(expressionInput      = multi_e,
             networkCoordinatesInput = hub_n,
             fileTypeInput       = "dat",
             geneSymbolInput     = "ID",
             readExpColumn       = readExpColumn("Cond_A-Cond_C",
                                                  "Cond_B-Cond_C"),
             resolutionValueInput = 10,
             smoothValueInput    = 5)
diff_res <- leviDiff(res2[[1]], res2[[2]])
diff_res$plot
```

leviEnrich

*leviEnrich***Description**

Run gene-set enrichment on the high-scoring (peaks) and low-scoring (valleys) regions returned by `levi()`. Uses `clusterProfiler` for GO and KEGG enrichment.

Usage

```
leviEnrich(
  result,
  top_n = 50L,
  bottom_n = 50L,
  organism = "hsa",
  orgdb = NULL,
  keytype = "SYMBOL",
  pval_cutoff = 0.05,
  types = c("GO_BP", "KEGG"),
  universe = NULL
)
```

Arguments

<code>result</code>	A single <code>levi()</code> result (list with <code>\$scores</code>).
<code>top_n</code>	Integer. Number of top-ranked genes (by landscape score) to test as the "over-expressed" gene set. Default is 50.
<code>bottom_n</code>	Integer. Number of bottom-ranked genes to test as the "under-expressed" gene set. Default is 50.
<code>organism</code>	Character. KEGG organism code (e.g., "hsa" for human, "mmu" for mouse). Also used to select the OrgDb for GO enrichment: pass the OrgDb package name as <code>orgdb</code> . See <code>clusterProfiler::enrichGO()</code> for details.
<code>orgdb</code>	Character. OrgDb package name for GO enrichment (e.g., "org.Hs.eg.db" for human). If NULL, GO enrichment is skipped.
<code>keytype</code>	Character. Key type of the gene identifiers in <code>result\$scores\$Gene</code> (e.g., "SYMBOL", "ENTREZID"). Default is "SYMBOL".
<code>pval_cutoff</code>	Numeric. Adjusted p-value threshold. Default 0.05.
<code>types</code>	Character vector. Which analyses to run. Any subset of <code>c("GO_BP", "GO_MF", "GO_CC", "KEGG")</code> . Default runs GO_BP and KEGG.
<code>universe</code>	Character vector. Background gene universe. If NULL (default), all genes in <code>result\$scores\$Gene</code> are used. For KEGG, both genes and universe are converted to ENTREZID using <code>orgdb</code> ; with <code>keytype = "ENTREZID"</code> , no conversion is needed.

Details

Requires clusterProfiler (Bioconductor). Install with `BiocManager::install("clusterProfiler")`. For GO enrichment also install the relevant OrgDb, e.g. `BiocManager::install("org.Hs.eg.db")`.

Value

A named list with elements over and under, each a named list of enrichment result objects (one per requested type). Plots a dot-plot for each non-empty result automatically.

Examples

```
hub_n <- system.file("extdata", "hub_network.dat", package = "levi")
hub_e <- system.file("extdata", "hub_expression.dat", package = "levi")
res <- levi(expressionInput      = hub_e,
             networkCoordinatesInput = hub_n,
             fileTypeInput       = "dat",
             geneSymbolInput     = "ID",
             readExpColumn       = readExpColumn("Test-Control"),
             resolutionValueInput = 10,
             smoothValueInput    = 5)

## Not run:
# Requires clusterProfiler and the org.Hs.eg.db annotation package,
# plus real gene symbols in the network.
enrich <- leviEnrich(res,
                     top_n      = 50,
                     orgdb     = "org.Hs.eg.db",
                     organism   = "hsa",
                     keytype    = "SYMBOL")

enrich$over$GO_BP
enrich$under$KEGG

## End(Not run)
```

leviFromBioc

leviFromBioc

Description

Universal adapter for Bioconductor expression containers. Automatically detects the class of obj and calls the appropriate `leviFrom*()` function.

Usage

```
leviFromBioc(obj, ...)
```

Arguments

- `obj` A Bioconductor expression object. Supported classes:
- SummarizedExperiment (and subclasses such as SingleCellExperiment or RangedSummarizedExperiment)
 - ExpressionSet (Biobase)
- `...` Additional arguments forwarded to [leviFromSE](#) or [leviFromExpressionSet](#).

Value

A data.frame with columns for gene identifiers, test, and control expression values, ready to pass to `levi()`.

See Also

[leviFromSE](#), [leviFromExpressionSet](#)

Examples

```
if (requireNamespace("SummarizedExperiment", quietly = TRUE)) {
  counts <- matrix(
    c(100, 50, 200, 120, 60, 210, 80, 150, 60),
    nrow = 3,
    dimnames = list(
      c("GENE1", "GENE2", "GENE3"),
      c("tumor1", "tumor2", "normal1")
    )
  )
  col_data <- data.frame(
    condition = c("Tumor", "Tumor", "Normal"),
    row.names = colnames(counts)
  )
  se <- SummarizedExperiment::SummarizedExperiment(
    assays = list(counts = counts),
    colData = col_data
  )
  expr_df <- leviFromBioc(se,
    condition_col = "condition",
    test_level = "Tumor",
    ctrl_level = "Normal")
  head(expr_df)
}
```

leviFromDESeq2

leviFromDESeq2

Description

Convert a DESeq2 result object into a data.frame compatible with `levi`'s `expressionInput` parameter.

Usage

```
leviFromDESeq2(dds_result, gene_col = "GeneID")
```

Arguments

`dds_result` A DESeqResults object returned by `DESeq2::results()`, or any `data.frame` with columns `baseMean` and `log2FoldChange`.

`gene_col` Character. Name for the gene identifier column. Default is "GeneID".

Details

Use `signal_mode = "logfc"` and `readExpColumn("log2FoldChange-log2FoldChange")`. The `baseMean` column is annotation, not a control measurement. Rows with NA in `log2FoldChange` are removed automatically.

Value

A `data.frame` with columns: gene identifier, `baseMean` (abundance annotation) and `log2FoldChange`, ready to pass to `levi(expressionInput = ...)`.

Examples

```
mock_res <- data.frame(
  baseMean      = c(120, 45, 80),
  log2FoldChange = c(2.1, -1.4, 0.2),
  row.names     = c("GENE1", "GENE2", "GENE3")
)
expr_df <- leviFromDESeq2(mock_res)
head(expr_df)
## Not run:
library(DESeq2)
res <- results(dds)
expr_df <- leviFromDESeq2(res, gene_col = "GeneSymbol")

## End(Not run)
```

leviFromEdgeR

leviFromEdgeR

Description

Convert an edgeR result object into a `data.frame` compatible with `levi`'s `expressionInput` parameter.

Usage

```
leviFromEdgeR(fit, coef = 1, gene_col = "GeneID")
```

Arguments

<code>fit</code>	A DGELRT or DGEEexact object from edgeR, or any data.frame with columns logFC and logCPM.
<code>coef</code>	Retained for call compatibility. Omit for edgeR test results; select the coefficient or contrast in glmLRT() or glmQLFTest() before calling this adapter.
<code>gene_col</code>	Character. Name for the gene identifier column. Default is "GeneID".

Value

A data.frame with columns: gene identifier, logCPM (abundance annotation) and logFC. Use `signal_mode = "logfc"` with `readExpColumn("logFC-logFC")`; abundance is not a control.

Examples

```
mock_tt <- data.frame(
  logFC = c(2.1, -1.4, 0.5),
  logCPM = c(5.2, 3.8, 4.1),
  row.names = c("GENE1", "GENE2", "GENE3")
)
expr_df <- leviFromEdgeR(mock_tt)
head(expr_df)
## Not run:
library(edgeR)
fit <- glmQLFit(dge, design)
qlf <- glmQLFTest(fit, coef = 2)
expr_df <- leviFromEdgeR(qlf, gene_col = "GeneSymbol")

## End(Not run)
```

leviFromEdges

*leviFromEdges***Description**

Build a levi-ready network from a plain list of interactions. `levi()` draws its landscape over the plane the network occupies, so it needs a coordinate for every node. When all you have is a list of pairs – which is what most interaction databases and most collaborators hand over – this computes a layout with **igraph** and returns the node and edge tables ready to pass straight to `levi()`.

Usage

```
leviFromEdges(
  edges,
  layout = c("fr", "kk", "lgl", "dh", "circle"),
  directed = FALSE,
  cols = seq_len(2L)
)
```

Arguments

<code>edges</code>	A <code>data.frame</code> or <code>matrix</code> whose first two columns name the endpoints of each interaction, or an igraph graph. Further columns are ignored.
<code>layout</code>	Character. Layout algorithm: <code>"fr"</code> (Fruchterman-Reingold, the default), <code>"kk"</code> (Kamada-Kawai), <code>"lgl"</code> , <code>"dh"</code> or <code>"circle"</code> .
<code>directed</code>	Logical. Whether to treat the pairs as directed when building the graph. The layout ignores direction either way. Default <code>FALSE</code> .
<code>cols</code>	Which two columns of edges hold the endpoints, by position or by name. Default is the first two.

Details

Force-directed layouts are stochastic: `"fr"`, `"kk"`, `"lgl"` and `"dh"` give a different arrangement on every call. Set `set.seed()` beforehand to reproduce a figure. `"circle"` is deterministic.

The layout is an analytical choice, not only an aesthetic one: the same expression values arranged differently produce a different landscape, because `levi` reads neighbourhoods. Where a curated layout exists, prefer it over a computed one.

Value

Invisibly, a list with:

nodes `data.frame` with name, x and y, the coordinates scaled to `[1, 100]`.

edges `data.frame` with V1 and V2, the endpoints.

graph the **igraph** object the layout was computed on.

Pass nodes to `networkCoordinatesInput`, edges to `networkInteractionsInput` and `"stg"` to `fileTypeInput`.

Author(s)

Jose Rybarczyk Filho (jose.luiz@unesp.br)

See Also

[levi](#), [leviFromSTRING](#)

Examples

```
# A small network given only as interactions, with no coordinates.
interactions <- data.frame(
  from = c("HUB", "HUB", "HUB", "HUB", "N1", "N2", "N3", "N4"),
  to   = c("N1", "N2", "N3", "N4", "N5", "N6", "N7", "N8"),
  stringsAsFactors = FALSE)

set.seed(42)
net <- leviFromEdges(interactions, layout = "kk")
head(net$nodes)
```

```

expression <- data.frame(
  ID      = c("HUB", paste0("N", 1:8)),
  Test    = c(200, 200, 200, 200, 200, 5, 5, 5, 5),
  Control = c(10, 10, 10, 10, 10, 200, 200, 200, 200))

res <- levi(
  networkCoordinatesInput = net$nodes,
  networkInteractionsInput = net$edges,
  fileTypeInput           = "stg",
  expressionInput          = expression,
  geneSymbolInput         = "ID",
  readExpColumn            = readExpColumn("Test-Control"),
  resolutionValueInput     = 20,
  smoothValueInput        = 50)

head(res$scores)

```

leviFromExpressionSet *leviFromExpressionSet*

Description

Convert an ExpressionSet (Biobase) object into a data.frame compatible with levi()'s expressionInput parameter. Typical use case: Bioconductor microarray datasets (Affymetrix, Agilent, etc.).

Usage

```

leviFromExpressionSet(
  eset,
  condition_col = NULL,
  test_level = NULL,
  ctrl_level = NULL,
  test_col = NULL,
  ctrl_col = NULL,
  gene_col = "GeneID"
)

```

Arguments

eset	An ExpressionSet object.
condition_col	Character. Column in pData(eset) that contains the condition labels. Used together with test_level and ctrl_level.
test_level	Character. Value in condition_col identifying the test samples.
ctrl_level	Character. Value in condition_col identifying the control samples.
test_col	Character vector. Sample names to use as test group (alternative to condition_col + test_level).
ctrl_col	Character vector. Sample names to use as control group.
gene_col	Character. Name for the gene identifier column in the output. Default "GeneID".

Details

Requires the Biobase package (Bioconductor). Install with `BiocManager::install("Biobase")`.

Two usage patterns:

1. **Condition labels:** specify `condition_col`, `test_level`, and `ctrl_level`.
2. **Sample names:** provide `test_col` and `ctrl_col` directly.

Value

A `data.frame` with columns `gene_col`, "Test", and "Control" containing per-gene mean intensity for each group. Pass directly to `levi(expressionInput = ...)`.

Examples

```
if (requireNamespace("Biobase", quietly = TRUE)) {
  mat <- matrix(c(5.1, 6.2, 4.8, 5.4, 6.0, 5.1, 7.3, 5.9, 4.2),
               nrow = 3,
               dimnames = list(
                 c("GENE1", "GENE2", "GENE3"),
                 c("tumor1", "tumor2", "normal1")))
  pd <- Biobase::AnnotatedDataFrame(data.frame(
    condition = c("Tumor", "Tumor", "Normal"),
    row.names = colnames(mat)))
  eset <- Biobase::ExpressionSet(assayData = mat, phenoData = pd)
  expr_df <- leviFromExpressionSet(eset,
    condition_col = "condition",
    test_level    = "Tumor",
    ctrl_level    = "Normal")
  head(expr_df)
}
```

leviFromLimma

leviFromLimma

Description

Convert a limma result object into a `data.frame` compatible with `levi`'s `expressionInput` parameter.

Usage

```
leviFromLimma(fit, coef = 1, gene_col = "GeneID")
```

Arguments

<code>fit</code>	A <code>MArrayLM</code> object after <code>limma::eBayes()</code> , or any <code>data.frame</code> with columns <code>logFC</code> and <code>AveExpr</code> .
<code>coef</code>	Integer or character. Coefficient to extract (used only when <code>fit</code> is a limma object). Default is 1.
<code>gene_col</code>	Character. Name for the gene identifier column. Default is "GeneID".

Value

A data.frame with columns: gene identifier, AveExpr (abundance annotation) and logFC. Use signal_mode = "logfc" with readExpColumn("logFC-logFC"); abundance is not a control.

Examples

```
mock_top <- data.frame(
  logFC = c(2.1, -1.4, 0.5),
  AveExpr = c(5.2, 3.8, 4.1),
  row.names = c("GENE1", "GENE2", "GENE3")
)
expr_df <- leviFromLimma(mock_top)
head(expr_df)
## Not run:
library(limma)
fit2 <- eBayes(fit)
expr_df <- leviFromLimma(fit2, coef = 1, gene_col = "GeneSymbol")

## End(Not run)
```

leviFromSE	<i>leviFromSE</i>
------------	-------------------

Description

Convert a SummarizedExperiment object into a data.frame compatible with levi's expressionInput parameter. Handles both single-assay and multi-assay SE objects.

Usage

```
leviFromSE(
  se,
  assay_name = "counts",
  test_col = NULL,
  ctrl_col = NULL,
  condition_col = NULL,
  test_level = NULL,
  ctrl_level = NULL,
  gene_col = "GeneID",
  log_transform = FALSE
)
```

Arguments

- se A SummarizedExperiment (or subclass) object.
- assay_name Character. Name or index of the assay to extract. Default is "counts" (falls back to assay index 1 if not found).

test_col	Character or integer. Column index/name in colData(se) identifying which samples are the test condition. Alternatively, a character vector of sample names.
ctrl_col	Character or integer. Column index/name in colData(se) identifying which samples are the control condition. Same format as test_col.
condition_col	Character. Column in colData(se) that contains condition labels. Used together with test_level / ctrl_level to select samples. Ignored when test_col / ctrl_col are provided directly.
test_level	Character. Value in condition_col identifying test samples.
ctrl_level	Character. Value in condition_col identifying control samples.
gene_col	Character. Name for the gene identifier column in the output. Default is "GeneID".
log_transform	Logical. If TRUE, apply $\log_2(x + 1)$ to the extracted counts before aggregating. Useful for count data. Default is FALSE.

Details

Two usage patterns:

1. **Direct sample selection:** provide sample names or indices via test_col / ctrl_col.
2. **Condition-label selection:** provide a colData column name via condition_col and the levels via test_level / ctrl_level.

Value

A data.frame with columns gene_col, "Test", and "Control" containing per-gene mean expression for each group. Pass directly to levi(expressionInput = ...).

Examples

```
if (requireNamespace("SummarizedExperiment", quietly = TRUE)) {
  counts <- matrix(
    c(100, 50, 200, 120, 60, 210, 80, 150, 60),
    nrow = 3,
    dimnames = list(
      c("GENE1", "GENE2", "GENE3"),
      c("tumor1", "tumor2", "normal1")
    )
  )
  col_data <- data.frame(
    condition = c("Tumor", "Tumor", "Normal"),
    row.names = colnames(counts)
  )
  se <- SummarizedExperiment::SummarizedExperiment(
    assays = list(counts = counts),
    colData = col_data
  )
  expr_df <- leviFromSE(se,
    condition_col = "condition",
    test_level = "Tumor",
    ctrl_level = "Normal")
}
```

```

      head(expr_df)
    }
    ## Not run:
    # Pattern 2: direct sample names, on your own SummarizedExperiment
    expr_df <- leviFromSE(se,
      test_col = c("tumor_1", "tumor_2"),
      ctrl_col = c("normal_1", "normal_2"))

    ## End(Not run)

```

leviFromSeurat

leviFromSeurat

Description

Convert a Seurat FindMarkers result into a data.frame compatible with levi's expressionInput parameter. Enables single-cell RNA-seq data visualization on biological networks.

Usage

```
leviFromSeurat(markers, gene_col = "GeneID")
```

Arguments

markers	A data.frame returned by Seurat::FindMarkers(), with columns avg_log2FC and pct.2.
gene_col	Character. Name for the gene identifier column. Default is "GeneID".

Details

Use signal_mode = "logfc" and readExpColumn("Test-Test"). The output Test column contains avg_log2FC; Control contains pct.2 as annotation, not a comparable control.

Value

A data.frame with columns: gene identifier, pct.2 renamed to Control (annotation) and avg_log2FC renamed to Test.

Examples

```

mock_markers <- data.frame(
  avg_log2FC = c(1.5, -0.8, 0.3),
  pct.2      = c(0.1, 0.4, 0.6),
  row.names  = c("GENE1", "GENE2", "GENE3")
)
expr_df <- leviFromSeurat(mock_markers)
head(expr_df)
## Not run:
library(Seurat)

```

```

markers <- FindMarkers(seurat_obj, ident.1 = "Tumor", ident.2 = "Normal")
expr_df <- leviFromSeurat(markers, gene_col = "GeneSymbol")
levi(expressionInput      = expr_df,
      networkCoordinatesInput = my_network,
      fileTypeInput       = "dat",
      geneSymbolInput     = "GeneSymbol",
      readExpColumn       = readExpColumn("Test-Test"),
      signal_mode         = "logfc")

## End(Not run)

```

leviFromSTRING

leviFromSTRING

Description

Build a levi-compatible network directly from the STRING protein interaction database using the **STRINGdb** Bioconductor package. Node coordinates are computed from the interaction graph via an igraph layout algorithm, so no manual network file download is required.

Usage

```

leviFromSTRING(
  genes,
  species = 9606,
  score_threshold = 400,
  network_type = c("full", "physical"),
  layout = c("fr", "kk", "lgl", "dh", "circle"),
  version = "11.5",
  input_directory = tempdir(),
  id_col = NULL
)

```

Arguments

genes	Character vector of gene symbols (or Entrez IDs / Ensembl IDs, matched by STRINGdb's mapping step). Genes not found in STRING are silently dropped and reported via message().
species	Integer. NCBI taxonomy ID. Common values: 9606 (human, default), 10090 (mouse), 10116 (rat), 7227 (Drosophila), 4932 (yeast), 6239 (C. elegans).
score_threshold	Integer (0-1000). Minimum STRING combined score to retain an interaction. 400 = medium confidence (default), 700 = high confidence, 900 = highest confidence.
network_type	Character. STRING network type: "full" (default, all channels) or "physical" (only physical protein-protein interactions).

layout	Character. igraph layout algorithm used to assign (x, y) coordinates to nodes. Options: "fr" Fruchterman-Reingold (default, good for most sizes), "kk" Kamada-Kawai (small networks, <=100 nodes), "lgl" Large Graph Layout (>500 nodes), "dh" Davidson-Harel, "circle".
version	Character. STRING database version. Default "11.5".
input_directory	Character. Local cache directory for STRINGdb files. Defaults to a temporary directory (re-downloaded each session). Set a persistent path (e.g. "~/stringdb_cache") to avoid re-downloading.
id_col	Character. Column in genes to map when genes is a data.frame. Ignored when genes is a character vector.

Details

Workflow:

1. STRINGdb maps gene symbols to STRING protein IDs.
2. Interactions above score_threshold are retrieved.
3. An igraph subnetwork is built and isolated nodes are retained.
4. A 2-D layout is computed (coordinates scaled to \[0, 100\]).
5. Gene symbols are recovered from the STRING node metadata.

Layout choice guide:

"fr"	Default. Force-directed, balanced for 50-500 nodes.
"kk"	Kamada-Kawai. Better aesthetics for <100 nodes.
"lgl"	Large Graph Layout. Use for >500 nodes.
"dh"	Davidson-Harel. Slow but high quality.
"circle"	Ring layout. Use for pathway-like linear chains.

Tip - seed for reproducibility: Force-directed layouts are stochastic. Set `set.seed()` before calling `leviFromSTRING()` when exact coordinates must be reproduced.

Value

A list with three elements:

`$nodes` data.frame with columns name (gene symbol), x, y - ready for `networkCoordinatesInput`.

`$edges` data.frame with columns V1, V2 (interacting gene symbols) - ready for `networkInteractionsInput`.

`$graph` The igraph object (invisible, for debugging or alternative layout experiments).

Pass `$nodes` and `$edges` directly to `levi()` with `fileTypeInput = "stg"`.

Author(s)

Jose Rafael Pilan (rafael.pilan@unesp.br)

Examples

```
# All examples below query the STRING database over the network and
# download a species-wide alias file (~20 MB), so they are not run
# during automated checks.
## Not run:
genes <- c("TP53", "BRCA1", "EGFR", "MYC", "PTEN")
set.seed(42)
net <- leviFromSTRING(genes, species = 9606, score_threshold = 700)
head(net$nodes)

# Human MAPK signalling genes
mapk_genes <- c("EGFR", "KRAS", "BRAF", "MAP2K1", "MAPK1", "MAPK3",
               "RPS6KA1", "MYC", "JUN", "FOS")
set.seed(42)
net <- leviFromSTRING(mapk_genes, species = 9606, score_threshold = 400)
levi(
  expressionInput      = my_expression_df, # your own data.frame
  networkCoordinatesInput = net$nodes,
  networkInteractionsInput = net$edges,
  fileTypeInput         = "stg",
  geneSymbolInput       = "GeneID",
  readExpColumn         = readExpColumn("Tumor-Normal"),
  signal_mode           = "logfc"
)

## End(Not run)
```

leviGraphClusterInference

Graph-connected cluster-mass inference from biological replicates

Description

Fits a moderated ‘limma’ t-statistic for every gene, thresholds the signed statistics, forms components using biological network edges, and assigns FWER-controlled P-values by permuting sample labels. The graph components, rather than adjacency on the 2D layout grid, are the inferential regions.

Usage

```
leviGraphClusterInference(
  expression,
  groups,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  test = unique(groups)[2],
```

```

control = unique(groups)[1],
threshold = 2,
n_perm = 999L,
blocks = NULL,
seed = NULL,
permutation_method = c("auto", "monte_carlo", "exact"),
max_exact = 50000L,
BPPARAM = SerialParam()
)

```

Arguments

expression	Log-scale genes-by-samples expression matrix.
groups	Two-level sample-group vector.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs.
test, control	Group labels defining test minus control.
threshold	Absolute moderated-t threshold for cluster formation.
n_perm	Number of sample-label permutations.
blocks	Optional exchangeability blocks.
seed	Optional RNG seed.
permutation_method	"auto", "monte_carlo" or "exact". "exact" enumerates every distinct label assignment; "monte_carlo" draws 'n_perm' random permutations; "auto" uses the exact scheme when the number of assignments is at most 'max_exact' and Monte Carlo otherwise.
max_exact	Maximum number of label assignments enumerated by the exact permutation scheme.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

List with observed moderated statistics, graph regions and null maxima.

Examples

```

if (requireNamespace("limma", quietly = TRUE)) {
  set.seed(1)
  genes <- c("HUB", paste0("N", 1:8))
  # Five paired donors: 32 within-donor label swaps, enumerated exactly.
  expression <- matrix(rnorm(length(genes) * 10, sd = .1), length(genes), 10,
    dimnames = list(genes, NULL))
  expression[, 6:10] <- expression[, 6:10] + 2
  out <- leviGraphClusterInference(expression, rep(c("C", "T"), each = 5),
    system.file("extdata", "hub_network.dat", package = "levi"),
    test = "T", control = "C", threshold = 1,

```

```

      blocks = rep(paste0("d", 1:5), 2), seed = 1)
    out$regions$summary
  }

```

leviGraphGetisOrd	<i>Getis-Ord hotspots and coldspots on a biological graph</i>
-------------------	---

Description

Computes the local Getis–Ord G_i^* statistic for every gene, including the node itself in its neighbourhood, and assigns one-sided node-label permutation P-values for hotspots and coldspots. A node adjacent to every other node has no variance to compare against and receives ‘NA’.

Usage

```

leviGraphGetisOrd(
  scores,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  n_perm = 999L,
  seed = NULL,
  BPPARAM = SerialParam()
)

```

Arguments

scores	Named numeric vector of gene scores (for example moderated t-statistics or log fold-changes) indexed by gene identifier. Network nodes without a finite score are dropped before testing.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
n_perm	Number of node-label permutations used to build the null.
seed	Optional RNG seed.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A data frame of local Getis–Ord G_i^* scores with one-sided permutation P-values for each direction, the two-sided P-value (twice the smaller one-sided value), its Benjamini-Hochberg adjustment across nodes and a ‘Significant’ flag at adjusted $P \leq 0.05$.

Examples

```
network <- data.frame(ID = c("A", "B", "C"), X = 1:3, Y = 1:3)
edges <- data.frame(V1 = c("A", "B"), V2 = c("B", "C"))
leviGraphGetisOrd(c(A = 1, B = 2, C = 3), network, edges,
  fileTypeInput = "stg", n_perm = 9)
```

leviGraphMoran

*Moran global and local autocorrelation on a biological graph***Description**

Computes global Moran's I and local Moran's I for gene scores using the binary adjacency of the biological network, with hotspot classes and node-label permutation P-values.

Usage

```
leviGraphMoran(
  scores,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  n_perm = 999L,
  seed = NULL,
  BPPARAM = SerialParam()
)
```

Arguments

scores	Named numeric vector of gene scores (for example moderated t-statistics or log fold-changes) indexed by gene identifier. Network nodes without a finite score are dropped before testing.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
n_perm	Number of node-label permutations used to build the null.
seed	Optional RNG seed.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A list with one global Moran's I test and a table of local Moran's I scores, hotspot classes, two-sided permutation P-values and Benjamini-Hochberg adjusted P-values across nodes.

Examples

```
network <- data.frame(ID = c("A", "B", "C"), X = 1:3, Y = 1:3)
edges <- data.frame(V1 = c("A", "B"), V2 = c("B", "C"))
leviGraphMoran(c(A = 1, B = 2, C = 3), network, edges,
  fileTypeInput = "stg", n_perm = 9)
```

leviGraphRewiringInference

Degree-preserving rewiring test for graph clusters

Description

Thresholds gene scores, forms connected components on the biological network and compares the observed maximum cluster mass with the mass obtained on degree-preserving rewired networks. This tests whether the clustering depends on the specific wiring rather than on the degree sequence.

Usage

```
leviGraphRewiringInference(
  scores,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  threshold = 2,
  n_perm = 999L,
  rewire_steps = NULL,
  seed = NULL,
  BPPARAM = SerialParam()
)
```

Arguments

scores	Named numeric vector of gene scores (for example moderated t-statistics or log fold-changes) indexed by gene identifier. Network nodes without a finite score are dropped before testing.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
threshold	Absolute score threshold for cluster formation.
n_perm	Number of rewired null networks.
rewire_steps	Number of degree-preserving edge swaps passed to [igraph::keeping_degseq()] for every null network. Defaults to ten times the number of edges (at least 10).
seed	Optional RNG seed.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A list of observed graph regions, degree-preserving rewiring null maxima and topology P-values.

Examples

```
network <- data.frame(ID = c("A", "B", "C", "D"), X = 1:4, Y = 1:4)
edges <- data.frame(V1 = c("A", "B", "C", "D"), V2 = c("B", "C", "D", "A"))
leviGraphRewiringInference(c(A = 3, B = 2.5, C = -1, D = -2), network, edges,
  fileTypeInput = "stg", n_perm = 9)
```

leviGraphSpectrum	<i>Laplacian smoothness and graph-Fourier low-frequency energy</i>
-------------------	--

Description

Measures how smoothly gene scores vary over the biological network through the Laplacian quadratic form and the share of signal energy in the lowest quarter of the graph-Fourier spectrum, with node-label permutation P-values. The spectral basis requires a full eigendecomposition of the Laplacian, computed once; the cost grows with the cube of the node count, so the test is intended for pathway-scale networks (up to a few thousand nodes).

Usage

```
leviGraphSpectrum(
  scores,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  n_perm = 999L,
  seed = NULL,
  BPPARAM = SerialParam()
)
```

Arguments

scores	Named numeric vector of gene scores (for example moderated t-statistics or log fold-changes) indexed by gene identifier. Network nodes without a finite score are dropped before testing.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
n_perm	Number of node-label permutations used to build the null.
seed	Optional RNG seed.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A one-row data frame of Laplacian energy, low-frequency energy ratio and node-label permutation P-values.

Examples

```
network <- data.frame(ID = c("A", "B", "C"), X = 1:3, Y = 1:3)
edges <- data.frame(V1 = c("A", "B"), V2 = c("B", "C"))
leviGraphSpectrum(c(A = 1, B = 2, C = 3), network, edges,
  fileTypeInput = "stg", n_perm = 9)
```

leviGraphTFCEFreedmanLane

Graph TFCE with Freedman–Lane residual permutation

Description

Tests a two-level condition while retaining nuisance covariates. Residuals from the nuisance-only model are permuted, then added back to fitted values before refitting the full model. ‘blocks’ restrict residual exchanges.

Usage

```
leviGraphTFCEFreedmanLane(
  expression,
  groups,
  covariates,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  test = unique(groups)[2],
  control = unique(groups)[1],
  blocks = NULL,
  n_perm = 999L,
  E = 0.5,
  H = 2,
  n_steps = 100L,
  seed = NULL,
  BPPARAM = SerialParam()
)
```

Arguments

expression	Log-scale genes-by-samples expression matrix with gene identifiers as row names.
groups	Two-level sample-group vector, one value per column.
covariates	Data frame of nuisance covariates with one row per sample. They define the reduced model whose residuals are permuted.

networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
test, control	Group labels defining the ‘test - control’ contrast.
blocks	Optional exchangeability blocks; residuals are exchanged only within blocks.
n_perm	Number of Freedman–Lane residual permutations.
E, H	TFCE extent and height exponents applied to the component size and to the threshold height at every integration step.
n_steps	Number of threshold steps used to integrate the TFCE score between zero and the maximum statistic.
seed	Optional RNG seed.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A list with gene-level statistics, TFCE scores and a global Freedman–Lane permutation null distribution.

Examples

```
network <- data.frame(ID = c("A", "B", "C"), X = 1:3, Y = 1:3)
edges <- data.frame(V1 = c("A", "B"), V2 = c("B", "C"))
expression <- matrix(rnorm(18), 3, dimnames = list(c("A", "B", "C"), NULL))
if (requireNamespace("limma", quietly = TRUE)) {
  leviGraphTFCEFreedmanLane(expression, rep(c("control", "case"), each = 3),
    data.frame(batch = rep(1:2, 3)), network, edges, fileTypeInput = "stg",
    n_perm = 19)
}
```

leviGraphTFCEInference

Threshold-free graph-cluster enhancement with sample-label inference

Description

Fits a moderated ‘limma’ t-statistic for every gene, integrates the signed statistics over all thresholds with threshold-free cluster enhancement (TFCE) using biological network components as clusters, and assigns FWER-controlled P-values by permuting sample labels. ‘PGlobal’ uses the joint maximum across genes and both directions.

Usage

```
leviGraphTFCEInference(
  expression,
  groups,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  test = unique(groups)[2],
  control = unique(groups)[1],
  blocks = NULL,
  n_perm = 999L,
  permutation_method = c("auto", "monte_carlo", "exact"),
  max_exact = 50000L,
  E = 0.5,
  H = 2,
  n_steps = 100L,
  seed = NULL,
  BPPARAM = SerialParam()
)
```

Arguments

expression	Log-scale genes-by-samples expression matrix with gene identifiers as row names.
groups	Two-level sample-group vector, one value per column.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
test, control	Group labels defining the 'test - control' contrast.
blocks	Optional exchangeability blocks (e.g. donor identifiers). Labels are permuted only within blocks and the block enters the linear model as a fixed effect.
n_perm	Number of Monte Carlo sample-label permutations.
permutation_method	"auto", "monte_carlo" or "exact". "exact" enumerates every distinct label assignment; "monte_carlo" draws 'n_perm' random permutations; "auto" uses the exact scheme when the number of assignments is at most 'max_exact' and Monte Carlo otherwise.
max_exact	Maximum number of label assignments enumerated by the exact permutation scheme.
E, H	TFCE extent and height exponents applied to the component size and to the threshold height at every integration step.
n_steps	Number of threshold steps used to integrate the TFCE score between zero and the maximum statistic.
seed	Optional RNG seed.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A list with signed gene statistics, TFCE scores, FWER-adjusted P-values and the permutation null maxima.

Examples

```
network <- data.frame(ID = c("A", "B", "C"), X = 1:3, Y = 1:3)
edges <- data.frame(V1 = c("A", "B"), V2 = c("B", "C"))
expression <- matrix(rnorm(18), 3, dimnames = list(c("A", "B", "C"), NULL))
if (requireNamespace("limma", quietly = TRUE)) {
  leviGraphTFCEInference(expression, rep(c("control", "case"), each = 3),
    network, edges, fileTypeInput = "stg", n_perm = 3)
}
```

leviGraphTFCERewiring *Degree-preserving rewiring sensitivity test for graph TFCE*

Description

Computes graph TFCE scores for fixed gene scores and compares them with the TFCE maxima obtained on degree-preserving rewired networks.

Usage

```
leviGraphTFCERewiring(
  scores,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  n_perm = 999L,
  rewire_steps = NULL,
  E = 0.5,
  H = 2,
  n_steps = 100L,
  seed = NULL,
  BPPARAM = SerialParam()
)
```

Arguments

scores	Named numeric vector of gene scores (for example moderated t-statistics or log fold-changes) indexed by gene identifier. Network nodes without a finite score are dropped before testing.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
n_perm	Number of rewired null networks.

rewire_steps	Number of degree-preserving edge swaps passed to [igraph::keeping_degseq()] for every null network. Defaults to ten times the number of edges (at least 10).
E, H	TFCE extent and height exponents applied to the component size and to the threshold height at every integration step.
n_steps	Number of threshold steps used to integrate the TFCE score between zero and the maximum statistic.
seed	Optional RNG seed.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A data frame with TFCE scores and degree-preserving rewiring P-values.

Examples

```
network <- data.frame(ID = c("A", "B", "C", "D"), X = 1:4, Y = 1:4)
edges <- data.frame(V1 = c("A", "B", "C", "D"), V2 = c("B", "C", "D", "A"))
leviGraphTFCERewiring(c(A = 3, B = 2.5, C = -1, D = -2), network, edges,
  fileTypeInput = "stg", n_perm = 9)
```

leviGraphWeightedTopology

Weighted Moran and Laplacian statistics for scored biological interactions

Description

‘edge_weights’ must have one value per interaction row (or be the name/index of a score column in an interaction data frame). It lets STRING confidence scores contribute to neighbourhood and smoothness calculations.

Usage

```
leviGraphWeightedTopology(
  scores,
  networkCoordinatesInput,
  networkInteractionsInput,
  edge_weights = 3L,
  fileTypeInput = "stg",
  n_perm = 999L,
  seed = NULL,
  BPPARAM = SerialParam()
)
```

Arguments

scores	Named numeric vector of gene scores (for example moderated t-statistics or log fold-changes) indexed by gene identifier. Network nodes without a finite score are dropped before testing.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs. 'networkInteractionsInput' must be supplied because the edge weights are read from its rows.
edge_weights	Positive edge weights, one per interaction row, or the name or index of the column in 'networkInteractionsInput' holding them (default: the third column).
n_perm	Number of node-label permutations used to build the null.
seed	Optional RNG seed.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A one-row data frame with weighted Moran's I, Laplacian energy and node-label permutation P-values.

Examples

```
network <- data.frame(ID = c("A", "B", "C"), X = 1:3, Y = 1:3)
edges <- data.frame(V1 = c("A", "B"), V2 = c("B", "C"), score = c(1, 2))
leviGraphWeightedTopology(c(A = 1, B = 2, C = 3), network, edges,
  edge_weights = "score", fileTypeInput = "stg", n_perm = 9)
```

leviGrid

*leviGrid***Description**

Arrange multiple levi() results side by side for visual comparison. Each panel shows one comparison's landscape at the same colour scale.

Usage

```
leviGrid(results, ncol = 2L, titles = NULL, ...)
```

Arguments

results	A levi result or a list of levi results (e.g., the output of a batch call with multiple readExpColumn() comparisons).
ncol	Integer. Number of columns in the grid. Default is 2.
titles	Character vector of panel titles. If NULL (default), uses the comparison field from each result.
...	Additional arguments passed to the layout engine.

Details

Requires one of the following packages (checked in order): patchwork, cowplot, or gridExtra.
Install any one with `install.packages()`.

Value

Invisibly returns the arranged plot object (class depends on the available package: patchwork, cowplot, or gridExtra).

Examples

```
hub_n <- system.file("extdata", "hub_network.dat", package = "levi")
hub_e <- system.file("extdata", "hub_expression.dat", package = "levi")
res <- levi(expressionInput      = hub_e,
             networkCoordinatesInput = hub_n,
             fileTypeInput       = "dat",
             geneSymbolInput     = "ID",
             readExpColumn       = readExpColumn("Test-Control"),
             resolutionValueInput = 10,
             smoothValueInput    = 5)
leviGrid(res)

# Batch mode: one landscape per comparison, arranged in a single column
multi_e <- system.file("extdata", "hub_multicomp_expression.dat",
                       package = "levi")
res2 <- levi(expressionInput      = multi_e,
             networkCoordinatesInput = hub_n,
             fileTypeInput       = "dat",
             geneSymbolInput     = "ID",
             readExpColumn       = readExpColumn("Cond_A-Cond_C",
                                                  "Cond_B-Cond_C"),
             resolutionValueInput = 10,
             smoothValueInput    = 5)
leviGrid(res2, ncol = 1)
```

leviPseudobulk

Aggregate single-cell counts to pseudobulks

Description

Aggregate single-cell counts to pseudobulks

Usage

```
leviPseudobulk(counts, donor, cell_type, condition = NULL, min_cells = 10L)
```

Arguments

counts	Genes-by-cells count matrix.
donor	Donor identifier per cell.
cell_type	Cell-type label per cell.
condition	Optional condition label per cell. When supplied, cells are aggregated separately for each donor-by-condition-by-cell-type combination.
min_cells	Minimum cells required for a pseudobulk.

Value

List with pseudobulk counts and donor, cell-type and (when supplied) condition metadata.

Examples

```
counts <- matrix(1:12, nrow = 3, dimnames = list(paste0("g", 1:3), NULL))
leviPseudobulk(counts, donor = c("d1", "d1", "d2", "d2"),
  cell_type = rep("T", 4), min_cells = 1)
```

leviRegionGenes	<i>Attribute a LEVI region to nearby network genes</i>
-----------------	--

Description

Ranks nodes by their Gaussian support inside each detected region, weighted by the node's departure from the neutral score. This is an interpretation score and not a gene-level P-value.

Usage

```
leviRegionGenes(result, top_n = 20L)
```

Arguments

result	A 'levi_result'.
top_n	Maximum genes returned per region; 'Inf' returns all.

Value

Data frame with regional gene contribution scores and ranks.

Examples

```

set.seed(1)
genes <- c("HUB", paste0("N", 1:8))
x <- levi(expressionInput = data.frame(ID = genes, logFC = rnorm(9, 1)),
  networkCoordinatesInput = system.file("extdata", "hub_network.dat", package = "levi"),
  fileTypeInput = "dat", geneSymbolInput = "ID",
  readExpColumn = readExpColumn("logFC-logFC"), signal_mode = "logfc",
  inference_unit = "region", n_perm = 3, resolutionValueInput = 1,
  region_threshold = .02)
leviRegionGenes(x, top_n = 3)

```

leviReplicateInference

Regional inference from biological replicates

Description

Permutes sample labels, recalculates gene log fold-changes, reconstructs the complete LEVI landscape and redetects regions at every iteration. This tests treatment association across biological replicates rather than node-label localisation on one fixed effect vector.

Usage

```

leviReplicateInference(
  expression,
  groups,
  gene_ids = rownames(expression),
  test = unique(groups)[2],
  control = unique(groups)[1],
  blocks = NULL,
  n_perm = 999L,
  sig_level = 0.05,
  seed = NULL,
  permutation_method = c("auto", "monte_carlo", "exact"),
  max_exact = 50000L,
  BPPARAM = SerialParam(),
  ...
)

```

Arguments

expression	Numeric genes-by-samples matrix on a log scale.
groups	Two-level condition vector, one value per sample.
gene_ids	Gene identifiers; defaults to row names.
test, control	Levels defining the ‘test - control’ contrast.
blocks	Optional permutation blocks (e.g. donor identifiers).

n_perm	Number of label permutations.
sig_level	Significance level.
seed	Optional RNG seed.
permutation_method	"auto", "monte_carlo" or "exact". "exact" enumerates every distinct label assignment; "monte_carlo" draws 'n_perm' random permutations; "auto" uses the exact scheme when the number of assignments is at most 'max_exact' and Monte Carlo otherwise.
max_exact	Maximum number of label assignments enumerated by the exact permutation scheme.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.
...	Network and landscape arguments passed to [levi()]. Do not supply expression, comparison, signal mode, permutation, or inference arguments.

Value

A 'levi_result' whose regional P-values are based on sample labels.

Examples

```
set.seed(1)
genes <- c("HUB", paste0("N", 1:8))
# Four replicates per group: 70 label arrangements, enumerated exactly.
expression <- matrix(rnorm(length(genes) * 8, sd = .1), length(genes), 8,
  dimnames = list(genes, NULL))
expression[, 5:8] <- expression[, 5:8] + 1
out <- leviReplicateInference(expression, rep(c("C", "T"), each = 4),
  test = "T", control = "C", seed = 2,
  networkCoordinatesInput = system.file("extdata", "hub_network.dat", package = "levi"),
  fileTypeInput = "dat", resolutionValueInput = 1, region_threshold = .02)
out$regions$summary
```

leviSave3D

Save the 3D landscape surface from a chosen viewpoint

Description

Writes the interactive surface returned by [levi](#) (field plot3d) to a file, optionally after setting the camera position, so that a figure can be reproduced from the same angle in every run. The format is taken from the file extension.

Usage

```
leviSave3D(
  x,
  file,
  camera = NULL,
  width = 1600,
  height = 1200,
  scale = 2,
  selfcontained = TRUE,
  compression = "LZW"
)
```

Arguments

<code>x</code>	A <code>levi_result</code> produced with <code>plot3d = TRUE</code> , or a plotly object such as <code>result\$plot3d</code> .
<code>file</code>	Output path. Supported extensions: <code>html</code> (interactive, keeps the view and can still be rotated), <code>png</code> , <code>jpeg</code> , <code>jpg</code> , <code>webp</code> , <code>svg</code> , <code>pdf</code> (static, written by <code>plotly::save_image</code> , which requires the Python packages <code>kaleido</code> and <code>plotly</code>) and <code>tiff/tif</code> (rendered as <code>PNG</code> and converted with the <code>magick</code> package).
<code>camera</code>	Optional camera specification, a list with any of <code>eye</code> , <code>center</code> and <code>up</code> , each a list with <code>x</code> , <code>y</code> and <code>z</code> . <code>eye</code> is the camera position relative to the centre of the surface; larger values move it away. <code>NULL</code> keeps the camera already stored in the plot (plotly's default view otherwise).
<code>width</code> , <code>height</code>	Image size in pixels for static formats.
<code>scale</code>	Resolution multiplier for static raster formats.
<code>selfcontained</code>	Logical; for <code>html</code> , embed all dependencies in a single file. Requires <code>pandoc</code> , and is turned off automatically when <code>pandoc</code> is not available.
<code>compression</code>	TIFF compression passed to <code>magick::image_write</code> ; <code>"LZW"</code> by default.

Details

Static export goes through `plotly::save_image`, which drives a headless browser via the `kaleido` and `plotly` Python packages. Install them once with `reticulate::py_install(c("kaleido", "plotly"))`. On this route a 1600 x 1200 export at scale 2 takes about ten seconds. Without them, save the HTML and use the camera icon of the plotly toolbar in a browser, or convert a PNG exported that way with `magick::image_write`.

Value

The path of the written file, invisibly.

See Also

[levi](#)

Examples

```

res <- levi(expressionInput = system.file("extdata", "expression.dat",
                                           package = "levi"),
            fileTypeInput = "dat",
            networkCoordinatesInput = system.file("extdata", "medusa.dat",
                                                    package = "levi"),
            geneSymbolInput = "ID",
            readExpColumn = readExpColumn("TumorCurrentSmoker-NormalNeverSmoker"),
            plot3d = TRUE)
cam <- list(eye = list(x = 1.6, y = -1.6, z = 0.9))
html <- tempfile(fileext = ".html")
leviSave3D(res, html, camera = cam, selfcontained = FALSE)
file.exists(html)

# Static formats need the kaleido and plotly Python packages
leviSave3D(res, tempfile(fileext = ".png"), camera = cam)
leviSave3D(res, tempfile(fileext = ".tiff"), camera = cam) # also magick

```

leviSingleCellGraphInference

Single-cell pseudobulk graph-cluster inference

Description

Aggregates cells by donor and cell type, then applies [leviBulkGraphInference] separately to each cell type. Donors, not cells, are permuted.

Usage

```

leviSingleCellGraphInference(
  counts,
  donor,
  cell_type,
  condition,
  min_cells = 10L,
  ...
)

```

Arguments

counts	Genes-by-cells raw counts.
donor, cell_type	Per-cell donor and cell-type labels.
condition	Named condition vector indexed by donor.
min_cells	Minimum cells required for each donor-cell-type pseudobulk; passed to [leviPseudobulk()].
...	Arguments forwarded to [leviBulkGraphInference].

Value

Named list of cell-type-specific graph inference results.

Examples

```
if (requireNamespace("DESeq2", quietly = TRUE)) {

  set.seed(1)
  genes <- c("HUB", paste0("N", 1:8))
  donor <- rep(paste0("d", 1:6), each = 4)
  cell_type <- rep(rep(c("A", "B"), each = 2), 6)
  counts <- matrix(rpois(length(genes) * length(donor), 30), length(genes),
    dimnames = list(genes, NULL))
  treated <- donor %in% c("d4", "d5", "d6")
  counts[, treated] <- counts[, treated] + 30L
  condition <- c(d1 = "C", d2 = "C", d3 = "C", d4 = "T", d5 = "T", d6 = "T")
  out <- leviSingleCellGraphInference(counts, donor, cell_type,
    condition = condition, min_cells = 2,
    networkCoordinatesInput = system.file("extdata", "hub_network.dat", package = "levi"),
    fileTypeInput = "dat", test = "T", control = "C", threshold = 1,
    n_perm = 3, seed = 1)
  names(out)

}
```

leviSingleCellInteractionTFCE

Cell-type-specific condition interaction with joint graph TFCE inference

Description

Tests whether the condition effect in each non-reference cell type differs from the reference type. Labels are exchanged within donors jointly over all cell types, avoiding cell-level pseudoreplication.

Usage

```
leviSingleCellInteractionTFCE(
  counts,
  donor,
  cell_type,
  condition,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  reference_cell_type = NULL,
  min_cells = 10L,
  cell_types = NULL,
```

```

n_perm = 999L,
permutation_method = c("auto", "monte_carlo", "exact"),
max_exact = 50000L,
E = 0.5,
H = 2,
n_steps = 100L,
seed = NULL,
normalize = c("TMM", "none"),
BPPARAM = SerialParam()
)

```

Arguments

counts	Genes-by-cells raw count matrix with gene row names.
donor, cell_type	One donor and cell-type label per cell.
condition	Two-level condition label per cell; every donor must contribute cells to both levels within each analysed cell type.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
reference_cell_type	Cell type against which the condition effect of every other type is contrasted. Defaults to the first complete type.
min_cells	Minimum cells in each donor-condition-cell-type pseudobulk.
cell_types	Optional cell types to analyse; at least two complete types are required.
n_perm	Number of Monte Carlo within-donor label permutations.
permutation_method	"auto", "monte_carlo" or "exact". "exact" enumerates every distinct label assignment; "monte_carlo" draws 'n_perm' random permutations; "auto" uses the exact scheme when the number of assignments is at most 'max_exact' and Monte Carlo otherwise.
max_exact	Maximum number of label assignments enumerated by the exact permutation scheme.
E, H	TFCE extent and height exponents applied to the component size and to the threshold height at every integration step.
n_steps	Number of threshold steps used to integrate the TFCE score between zero and the maximum statistic.
seed	Optional RNG seed.
normalize	Library-size normalisation applied before the log2 counts-per-million transform: "TMM" (trimmed mean of M-values via edgeR , the default) or "none" (plain counts per million).
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A list of interaction TFCE tables and the joint donor-label null distribution.

Examples

```
if (requireNamespace("limma", quietly = TRUE)) {
  set.seed(1)
  genes <- c("HUB", paste0("N", 1:8))
  donor <- rep(paste0("d", 1:5), each = 4)
  condition <- rep(rep(c("ctrl", "stim"), each = 2), 5)
  cell_type <- rep(c("A", "B"), times = 10)
  counts <- matrix(rpois(length(genes) * length(donor), 20), length(genes),
    dimnames = list(genes, NULL))
  out <- leviSingleCellInteractionTFCE(counts, donor, cell_type, condition,
    system.file("extdata", "hub_network.dat", package = "levi"),
    reference_cell_type = "A", min_cells = 1, n_perm = 19,
    permutation_method = "monte_carlo", seed = 1)
  out$results$B
}
```

levisingleCellRegionalInference

Joint regional inference for paired single-cell pseudobulks

Description

Computes graph-connected cluster mass and explicit two-dimensional LEVI areas for every selected cell type. Treatment labels are swapped within donors and the null stores the maximum region mass across cell types, directions and regions, controlling the complete search family.

Usage

```
levisingleCellRegionalInference(
  counts,
  donor,
  cell_type,
  condition,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  min_cells = 10L,
  cell_types = NULL,
  threshold = 2,
  n_perm = 999L,
  permutation_method = c("auto", "monte_carlo", "exact"),
  max_exact = 50000L,
  seed = NULL,
  normalize = c("TMM", "none"),
```

```

    BPPARAM = SerialParam(),
    ...
)

```

Arguments

counts	Genes-by-cells raw count matrix with gene row names.
donor, cell_type	One donor and cell-type label per cell.
condition	Two-level condition label per cell; every donor must contribute cells to both levels within each analysed cell type.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
min_cells	Minimum cells in each donor-condition-cell-type pseudobulk.
cell_types	Optional cell types to analyse. By default, types with a complete donor-by-condition pseudobulk set are used.
threshold	Absolute moderated-t threshold for graph cluster formation.
n_perm	Number of Monte Carlo within-donor label permutations.
permutation_method	"auto", "monte_carlo" or "exact". "exact" enumerates every distinct label assignment; "monte_carlo" draws 'n_perm' random permutations; "auto" uses the exact scheme when the number of assignments is at most 'max_exact' and Monte Carlo otherwise.
max_exact	Maximum number of label assignments enumerated by the exact permutation scheme.
seed	Optional RNG seed.
normalize	Library-size normalisation applied before the log2 counts-per-million transform: "TMM" (trimmed mean of M-values via edgeR , the default) or "none" (plain counts per million).
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.
...	Landscape arguments passed to [levi()] (for example 'resolutionValueInput' or 'region_threshold'). Do not supply expression, network, signal mode, permutation or inference arguments.

Value

A list of graph-cluster and two-dimensional landscape regional tests.

Examples

```

if (requireNamespace("limma", quietly = TRUE)) {
  set.seed(1)
  genes <- c("HUB", paste0("N", 1:8))
  # Five paired donors: 32 within-donor swaps; 19 Monte Carlo draws here

```

```

# keep the example fast.
donor <- rep(paste0("d", 1:5), each = 4)
condition <- rep(rep(c("ctrl", "stim"), each = 2), 5)
cell_type <- rep(c("A", "B"), times = 10)
counts <- matrix(rpois(length(genes) * length(donor), 20), length(genes),
  dimnames = list(genes, NULL))
out <- leviSingleCellRegionalInference(counts, donor, cell_type, condition,
  system.file("extdata", "hub_network.dat", package = "levi"),
  min_cells = 1, threshold = 1, n_perm = 19,
  permutation_method = "monte_carlo", seed = 1,
  resolutionValueInput = 1, region_threshold = .02)
out$cluster_mass$A$summary
}

```

levisingleCellTFCEInference

Cell-type-specific TFCE inference for pseudobulk single-cell RNA-seq

Description

Aggregates raw counts by donor and cell type, computes a limma-trend moderated t statistic per gene and cell type on TMM-normalised log2 CPM, then uses graph TFCE. Donor labels are permuted jointly across all selected cell types. ‘PGlobal’ controls the search over genes, directions, regions and cell types.

Usage

```

levisingleCellTFCEInference(
  counts,
  donor,
  cell_type,
  condition,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  min_cells = 10L,
  cell_types = NULL,
  n_perm = 999L,
  permutation_method = c("auto", "monte_carlo", "exact"),
  max_exact = 50000L,
  E = 0.5,
  H = 2,
  n_steps = 100L,
  seed = NULL,
  normalize = c("TMM", "none"),
  BPPARAM = SerialParam()
)

```

Arguments

counts	Genes-by-cells raw count matrix.
donor, cell_type	One donor and cell-type label per cell.
condition	Either a named two-level vector indexed by donor (a between-donor comparison) or one two-level label per cell (a paired or repeated-condition comparison). In the latter case, labels are permuted within donor.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
min_cells	Minimum cells in each donor-cell-type pseudobulk.
cell_types	Optional cell types to analyse. By default, types present in every donor are used, ensuring joint donor permutations are valid.
n_perm	Number of Monte Carlo donor-label permutations.
permutation_method	"auto", "monte_carlo" or "exact". "exact" enumerates every distinct label assignment; "monte_carlo" draws 'n_perm' random permutations; "auto" uses the exact scheme when the number of assignments is at most 'max_exact' and Monte Carlo otherwise.
max_exact	Maximum number of label assignments enumerated by the exact permutation scheme.
E, H	TFCE extent and height exponents applied to the component size and to the threshold height at every integration step.
n_steps	Number of threshold steps used to integrate the TFCE score between zero and the maximum statistic.
seed	Optional RNG seed.
normalize	Library-size normalisation applied before the log2 counts-per-million transform: "TMM" (trimmed mean of M-values via edgeR , the default) or "none" (plain counts per million).
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A list of cell-type tables, global null maxima and metadata.

Examples

```
if (requireNamespace("limma", quietly = TRUE)) {
  set.seed(1)
  genes <- c("HUB", paste0("N", 1:8))
  # Six donors, three per condition: 20 donor-label arrangements.
  donor <- rep(paste0("d", 1:6), each = 4)
  cell_type <- rep(rep(c("A", "B"), each = 2), 6)
  counts <- matrix(rpois(length(genes) * length(donor), 10), length(genes),
```

```

    dimnames = list(genes, NULL))
  condition <- c(d1 = "C", d2 = "C", d3 = "C", d4 = "T", d5 = "T", d6 = "T")
  out <- leviSingleCellTFCEInference(counts, donor, cell_type, condition,
    system.file("extdata", "hub_network.dat", package = "levi"),
    min_cells = 2, seed = 1)
  out$results$A
}

```

leviSingleCellTopologyInference

Run Moran, Laplacian/Fourier and rewiring tests per cell type

Description

Takes the moderated t-statistics stored in a [leviSingleCellTFCEInference()] result and applies [leviGraphMoran()], [leviGraphSpectrum()] and [leviGraphRewiringInference()] to every cell type.

Usage

```

leviSingleCellTopologyInference(
  single_cell_tfce,
  networkCoordinatesInput,
  networkInteractionsInput = NA,
  fileTypeInput = "dat",
  threshold = 2,
  n_perm = 999L,
  seed = NULL,
  BPPARAM = SerialParam()
)

```

Arguments

single_cell_tfce	Result of [leviSingleCellTFCEInference()]; the ‘T’ column of every cell-type table supplies the gene scores.
networkCoordinatesInput, networkInteractionsInput, fileTypeInput	LEVI network inputs: node coordinates, optional interactions and the file type accepted by [levi()].
threshold	Absolute t-statistic threshold for cluster formation in the rewiring test.
n_perm	Number of node-label permutations or rewired networks used by each topology test.
seed	Optional RNG seed.
BPPARAM	A [BiocParallel::BiocParallelParam] object controlling how the permutations are evaluated. The default runs them serially.

Value

A named list of topology-test results, one entry for each cell type.

Examples

```
if (requireNamespace("limma", quietly = TRUE)) {
  set.seed(1)
  net <- system.file("extdata", "hub_network.dat", package = "levi")
  genes <- c("HUB", paste0("N", 1:8))
  donor <- rep(paste0("d", 1:6), each = 4)
  cell_type <- rep(rep(c("A", "B"), each = 2), 6)
  counts <- matrix(rpois(length(genes) * length(donor), 10), length(genes),
    dimnames = list(genes, NULL))
  condition <- c(d1 = "C", d2 = "C", d3 = "C", d4 = "T", d5 = "T", d6 = "T")
  tfce <- leviSingleCellTFCEInference(counts, donor, cell_type, condition,
    net, min_cells = 2, seed = 1)
  topology <- leviSingleCellTopologyInference(tfce, net, threshold = 1,
    n_perm = 19, seed = 1)
  topology$A$moran$global
}
```

 LEVIui

LEVIui - Interactive Shiny GUI for levi

Description

Launch the **levi** graphical user interface (GUI) locally via Shiny. The GUI exposes all parameters available in script mode and additionally provides:

- Interactive brushable region selection on the landscape heatmap (returns gene scores for the selected area).
- Gene search - type a gene symbol to highlight its position with a yellow circle on the landscape.
- Peak label overlay (*requires* **ggrepel**).
- Signal transformation mode selector (ratio, logfc, zscore) and logfc_k steepness control.
- Interactive 3D surface (*requires* **plotly**), carrying the same significance boundary as the 2D map.
- Permutation significance test with per-iteration progress bar.
- Download buttons for the landscape plot, node score table, and peak/valley table (CSV).

Usage

```
LEVIui(browser = FALSE)
```

Arguments

browser	Logical. TRUE opens the app in the system web browser; FALSE (default) opens it in the RStudio Viewer pane.
---------	---

Details

The GUI is a full-featured interface to [levi](#). All file uploads, parameter sliders, and result downloads are handled interactively. For reproducible, automated, or batch analyses use [levi](#) directly in script mode.

The app is located in `inst/shiny/` and can also be launched with `shiny::runApp(system.file("shiny", package = "levi"))`.

Value

Runs the Shiny application; does not return an R value.

Author(s)

Jose Rafael Pilan <rafael.pilan@unesp.br>, Isabelle Mira da Silva

See Also

[levi](#), [readExpColumn](#)

Examples

```
if (interactive()) {
  LEVIui(browser = FALSE) # opens in RStudio Viewer
  LEVIui(browser = TRUE)  # opens in system browser
}
```

logfc_dataset

Log2-intensity (logfc-mode) toy dataset

Description

A 6-node linear chain identical in topology to the [gradient dataset](#), but with expression values representing log2 microarray intensities (range 5.0-8.0). The logFC between test and control spans -3 to +3 log2 units. Designed to show that `signal_mode = "logfc"` extracts wider contrast than `signal_mode = "ratio"` on log-scale data.

Format

Two tab-delimited files in `inst/extdata/`:

`logfc_network.dat` Linear chain GA-GF, identical layout to `gradient_network.dat`.

`logfc_expression.dat` Columns ID, Test, Control in log2 intensity units:

Gene	Test	Control
GA	5.0	8.0
GB	6.0	7.5

GC	7.0	7.5
GD	7.5	7.0
GE	8.0	6.5
GF	8.0	5.0

LogFC = Test - Control ranges from -3 (GA) to +3 (GF).

Value

NULL (this object documents data files, not a function)

Signal mode comparison

signal_mode = "logfc" sigmoid(-3) ~ 0.047 (GA) -> sigmoid(+3) ~ 0.953 (GF). Score range ~ 0.91 - strong biological contrast.

signal_mode = "ratio" $5/(5+8) \sim 0.38$ (GA) -> $8/(8+5) \sim 0.62$ (GF). Score range ~ 0.24 - weaker contrast from treating log2 values as linear.

Use signal_mode = "logfc" whenever Test and Control represent log2-transformed intensities (RMA microarray, VST/rlog, log2-proteomics).

Source

Simulated log2 microarray intensities for unit testing.

See Also

[levi](#), [gradient dataset](#)

Examples

```
logfc_net <- file.path(system.file(package = "levi"),
                        "extdata", "logfc_network.dat")
logfc_expr <- file.path(system.file(package = "levi"),
                        "extdata", "logfc_expression.dat")

# logfc mode - recommended for log2-scale data
res_lfc <- levi(
  networkCoordinatesInput = logfc_net,
  expressionInput         = logfc_expr,
  fileTypeInput           = "dat",
  geneSymbolInput         = "ID",
  readExpColumn           = readExpColumn("Test-Control"),
  signal_mode             = "logfc",
  contrastValueInput      = 50,
  resolutionValueInput    = 50,
  zoomValueInput          = 50,
  smoothValueInput        = 50
)
diff(range(res_lfc$scores$LandscapeScore)) # score range with logfc mode
```

multicomp_dataset	<i>Multi-comparison toy dataset</i>
-------------------	-------------------------------------

Description

A three-condition expression file for the [hub network](#), enabling batch-mode processing and testing of [leviGrid](#) and [leviDiff](#). Condition A is the maximum-contrast state (hub up, corners down); Condition B is neutral; Condition C is the reverse of A.

Format

One tab-delimited file in inst/extdata/ with columns ID, Cond_A, Cond_B, Cond_C:

Gene	Cond_A	Cond_B	Cond_C
HUB	200	100	10
N1-N4	200	100	10
N5-N8	10	100	200

Designed to be used with `hub_network.dat` (`fileTypeInput = "dat"`).

Value

NULL (this object documents data files, not a function)

Use cases

- Batch mode** `readExpColumn("Cond_A-Cond_B", "Cond_A-Cond_C")` returns two landscapes in a single `levi()` call.
- leviGrid** Compare the two landscapes side by side.
- leviDiff** Subtract `Cond_A-Cond_B` from `Cond_A-Cond_C` to reveal the additional contrast gained by Condition C.

Source

Simulated data generated for unit testing.

See Also

[levi](#), [leviGrid](#), [leviDiff](#)

Examples

```
hub_net <- file.path(system.file(package = "levi"),
                     "extdata", "hub_network.dat")
mc_expr <- file.path(system.file(package = "levi"),
                     "extdata", "hub_multicomp_expression.dat")

res_list <- levi(
  networkCoordinatesInput = hub_net,
  expressionInput         = mc_expr,
  fileTypeInput           = "dat",
  geneSymbolInput         = "ID",
  readExpColumn           = readExpColumn("Cond_A-Cond_B",
                                           "Cond_A-Cond_C")
)
leviGrid(res_list, titles = c("A vs B", "A vs C"))
leviDiff(res_list[[1]], res_list[[2]])
```

```
print.levi_result      Print a levi result
```

Description

Compact summary of the object returned by [levi](#): the comparison, how many nodes were scored, the strongest and weakest genes, the peaks and valleys detected, and whether a permutation test was run.

Usage

```
## S3 method for class 'levi_result'
print(x, n = 3L, ...)
```

Arguments

x	A <code>levi_result</code> object, as returned by levi .
n	Integer. How many genes to list at each end of the ranking. Default is 3.
...	Ignored, present for compatibility with the generic.

Details

The landscape plot is drawn by [levi](#) itself, so this method prints text only and never redraws the figure. Use `x$plot` to get the `ggplot` object back.

Value

x, invisibly. Called for the summary it prints.

See Also

[levi](#)

Examples

```
hub_n <- system.file("extdata", "hub_network.dat", package = "levi")
hub_e <- system.file("extdata", "hub_expression.dat", package = "levi")

res <- levi(networkCoordinatesInput = hub_n,
            expressionInput          = hub_e,
            fileTypeInput            = "dat",
            geneSymbolInput          = "ID",
            readExpColumn             = readExpColumn("Test-Control"),
            resolutionValueInput      = 10,
            smoothValueInput          = 5)

res
```

readExpColumn	<i>readExpColumn - Define expression comparisons for levi</i>
---------------	---

Description

Specify which columns of the expression file represent the *test* condition and the *control* condition for the landscape computation. Multiple comparisons can be provided in a single call to enable batch-mode processing (one landscape per comparison).

Usage

```
readExpColumn(x, ...)
```

Arguments

- | | |
|-----|--|
| x | Character string of the form "TestColumn-ControlColumn", where TestColumn and ControlColumn are column names in the expression file. The hyphen (-) separates test from control. Special cases: <ul style="list-style-type: none">• Single-column logFC input (e.g., DESeq2 log2FoldChange only): repeat the same column name on both sides - "log2FoldChange-log2FoldChange".• Self-comparison (null landscape): use the same column twice - "Sample-Sample". |
| ... | Additional comparisons, each following the same "Test-Control" format. When multiple comparisons are supplied, levi() generates one landscape per comparison and returns a list. |

Details

Column names must match exactly the column headers in the expression file or data.frame (case-sensitive). The hyphen separator is required; column names that themselves contain hyphens should be renamed before use.

When using `signal_mode = "logfc"` with a single-column fold-change input (e.g., from `leviFromDESeq2`), repeat the logFC column name: `readExpColumn("log2FoldChange-log2FoldChange")`. The function detects `baseTest == baseControl` and routes to single-column mode automatically.

Value

A named list used internally by `levi` to identify the expression columns for each comparison. The first element is always the function call itself (used as a sentinel); subsequent elements are the comparison strings.

Author(s)

Jose Rafael Pilan <rafael.pilan@unesp.br>

See Also

`levi`, `leviFromDESeq2`, `leviFromEdgeR`, `leviFromLimma`, `leviFromSeurat`

Examples

```
# Single comparison
readExpColumn("TumorCurrentSmoker-NormalNeverSmoker")

# Multiple comparisons (batch mode) - levi() returns a list
readExpColumn(
  "TumorCurrentSmoker-NormalNeverSmoker",
  "TumorFormerSmoker-NormalFormerSmoker"
)

# Single-column logFC (repeat the column name)
readExpColumn("log2FoldChange-log2FoldChange")

# Self-comparison (null landscape, all scores ~ 0.5)
readExpColumn("Sample1-Sample1")
```

sparse_dataset

Sparse-expression toy dataset

Description

A 15-node linear chain where only 5 of the 15 network genes have measured expression values. The remaining 10 genes are absent from the expression file and receive the mode-appropriate neutral value (0.5 for `signal_mode = "ratio"`; 0.0 for `"logfc"` and `"zscore"`). Used to verify that missing genes are handled gracefully and that scores are returned for all 15 network nodes.

Format

Two tab-delimited files in inst/extdata/:

`sparse_network.dat` Linear chain S01-S15 with 14 edges, nodes equally spaced from $x = 0.05$ to $x = 0.95$ at $y = 0.50$.

`sparse_expression.dat` Only genes S01-S05 are present. Expression spans from strongly up-regulated (S01: Test = 200, Control = 10) to strongly down-regulated (S05: Test = 10, Control = 200). Genes S06-S15 are not listed and receive neutral imputation.

Value

NULL (this object documents data files, not a function)

Expected results

- `nrow(result$scores) = 15` (all network nodes are scored).
- `score(S01) > score(S05)`.
- No error or warning is raised for the 10 missing genes.
- Both `signal_mode = "ratio"` and `"zscore"` complete successfully; missing-gene neutral values differ between modes but the outcome is valid for both.

Source

Simulated data generated for unit testing.

See Also

[levi](#)

Examples

```
sparse_net <- file.path(system.file(package = "levi"),
                        "extdata", "sparse_network.dat")
sparse_expr <- file.path(system.file(package = "levi"),
                        "extdata", "sparse_expression.dat")

res <- levi(
  networkCoordinatesInput = sparse_net,
  expressionInput         = sparse_expr,
  fileTypeInput           = "dat",
  geneSymbolInput         = "ID",
  readExpColumn           = readExpColumn("Test-Control"),
  contrastValueInput      = 50,
  resolutionValueInput    = 50,
  zoomValueInput          = 50,
  smoothValueInput        = 50
)
nrow(res$scores) # 15 - all network nodes scored
```

Index

bimodal_dataset, 3
bimodal_expression (bimodal_dataset), 3
bimodal_network (bimodal_dataset), 3

flat_dataset, 4
flat_expression (flat_dataset), 4
flat_network (flat_dataset), 4

gradient_dataset, 5
gradient_expression (gradient_dataset), 5
gradient_network (gradient_dataset), 5

hub_dataset, 7
hub_expression (hub_dataset), 7
hub_multicomp_expression (multicomp_dataset), 59
hub_network (hub_dataset), 7

levi, 4–7, 8, 23, 46, 47, 57–63
leviAdjustPathways, 13
leviBulkGraphInference, 14
leviDiff, 3, 4, 12, 16, 59
leviEnrich, 18
leviFromBioc, 19
leviFromDESeq2, 20, 62
leviFromEdgeR, 21, 62
leviFromEdges, 22
leviFromExpressionSet, 20, 24
leviFromLimma, 25, 62
leviFromSE, 20, 26
leviFromSeurat, 28, 62
leviFromSTRING, 23, 29
leviGraphClusterInference, 31
leviGraphGetisOrd, 33
leviGraphMoran, 34
leviGraphRewiringInference, 35
leviGraphSpectrum, 36
leviGraphTFCEFreemanLane, 37
leviGraphTFCEInference, 38
leviGraphTFCERewiring, 40
leviGraphWeightedTopology, 41
leviGrid, 42, 59
leviPseudobulk, 43
leviRegionGenes, 44
leviReplicateInference, 45
leviSave3D, 46
leviSingleCellGraphInference, 48
leviSingleCellInteractionTFCE, 49
leviSingleCellRegionalInference, 51
leviSingleCellTFCEInference, 53
leviSingleCellTopologyInference, 55
LEVIui, 56
logfc_dataset, 57
logfc_expression (logfc_dataset), 57
logfc_network (logfc_dataset), 57

multicomp_dataset, 59

print.levi_result, 60

readExpColumn, 57, 61

sparse_dataset, 62
sparse_expression (sparse_dataset), 62
sparse_network (sparse_dataset), 62