

Package: hcocena (via r-universe)

October 4, 2026

Title Horizontal Integration and Analysis of Multiple Transcriptomics Datasets

Version 0.99.12

Description Tools for the joint analysis of multiple transcriptomics datasets using a network-based integration approach. The package provides an S4 object model based on MultiAssayExperiment for reproducible analysis of single-layer and multi-layer studies. It includes workflows for clustering, module visualization, functional annotation, upstream inference, and longitudinal follow-up while retaining compatibility helpers for legacy hCoCena objects.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.5.0)

biocViews Software, Clustering, GeneExpression, KEGG, Network, NetworkInference, Transcriptomics, WorkflowStep

BiocType Software

URL <https://github.com/BioCompNet/hcocena>

BugReports <https://github.com/BioCompNet/hcocena/issues>

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports Cairo (>= 1.5-12.2), circlize, clusterProfiler (>= 3.18.1), ComplexHeatmap (>= 2.6.2), cowplot (>= 1.1.1), dendextend (>= 1.15.1), dplyr (>= 1.0.6), factoextra (>= 1.0.7), ggplot2 (>= 3.3.5), ggsci (>= 2.9), graphics (>= 4.0.3), grDevices (>= 4.0.3), grid (>= 4.0.3), gtools (>= 3.9.2), Hmisc (>= 4.5-0), httr (>= 1.4.2), igraph (>= 1.2.6), jsonlite (>= 1.7.2), leidenAlg (>= 0.1.1), leidenbase (>= 0.1.25), magrittr (>= 2.0.1), Matrix, methods, MultiAssayExperiment, openxlsx (>= 4.2.4), patchwork (>= 1.1.3), pheatmap (>= 1.0.12), plotly (>= 4.9.4.1), purrr (>= 0.3.4), RColorBrewer (>= 1.1-2), readr (>= 2.0.1), reshape2, rlist (>= 0.4.6.1), S4Vectors, scales (>=

1.1.1), stringi (>= 1.6.2), stringr (>= 1.5.1),
 SummarizedExperiment, tibble (>= 3.1.2), tidyselect (>= 1.1.1),
 zip (>= 2.3.0)

Suggests BiocStyle, CALIBERrfimpute, clipr, decoupleR, dorothea,
 ellmer, emmeans, ggh4x, ggrepel, graphlayouts, GSVA,
 htmlwidgets, knitr (>= 1.33), lme4, limma, mice, networkD3 (>= 0.4), nlme, OmnipathR, png, progeny, RANN, RCy3 (>= 2.10.2),
 RcppAnnoy, repr, rmarkdown, SingleCellExperiment, testthat (>= 3.0.0), umap, uwot, withr

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev perl libx11-dev zlib1g-dev

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

Date/Publication 2026-10-03 21:41:46 UTC

RemoteUrl <https://github.com/BiocStaging/hcocena>

RemoteRef HEAD

RemoteSha d0b918c0e0b390678cceba94ccaa9d2fc0ddd95d

Contents

hc_add_meta_time_grouping	5
hc_algo_alluvial	6
hc_antilog	7
hc_auto_set_paths	7
hc_auto_tune	9
hc_auto_tune_cutoff_tiered	11
hc_build_integrated_network	12
hc_celltype_activity_decoupler	13
hc_celltype_annotation	15
hc_change_grouping_parameter	17
hc_check_dirs	18
hc_check_tf	19
hc_cluster_calculation	20
hc_clusters	21
hc_col_anno_categorical	22
hc_col_anno_numerical	22
hc_colour_single_cluster	23
hc_config	24
hc_cut_hclust	24
hc_define_layers	25
hc_example_data	26
hc_export_clusters	27
hc_export_to_cytoscape	27

hc_export_to_local_folder	28
hc_find_hubs	29
hc_functional_enrichment	30
hc_gene_to_cluster	34
hc_get_meta_time_col_order	34
hc_get_module_scores	35
hc_graph	36
hc_highlight_geneset	36
hc_import_clusters	37
hc_import_layout_from_cytoscape	38
hc_import_layout_from_local_folder	39
hc_init	40
hc_init_save_folder	40
hc_integration	41
hc_layer_results	41
hc_list_celltype_databases	42
hc_list_llm_models	43
hc_llm_enrichment	44
hc_llm_import	47
hc_longitudinal_endotype_clustering	48
hc_longitudinal_meta_clustering	49
hc_longitudinal_module_cap	51
hc_longitudinal_module_means	53
hc_longitudinal_step1_module_donor	54
hc_longitudinal_step1_module_donor_direct	56
hc_longitudinal_step2_meta_clustering	58
hc_longitudinal_step3_meta_module_trajectories	59
hc_longitudinal_workflow_direct	61
hc_mae	63
hc_merge_clusters	63
hc_meta_correlation_cat	64
hc_meta_correlation_num	65
hc_meta_plot	66
hc_module_condition_significance	67
hc_module_gfc_tables	68
hc_network_comparison_1	70
hc_network_comparison_2	71
hc_pca	72
hc_pca_algo_compare	73
hc_plot_celltype_annotation_matrix	73
hc_plot_cluster_heatmap	75
hc_plot_cluster_heatmap_view	75
hc_plot_cutoffs	77
hc_plot_deg_dist	78
hc_plot_enrichment_panels	78
hc_plot_enrichment_upstream_network	79
hc_plot_gfc_network	82
hc_plot_integrated_network	82

hc_plot_llm_enrichment	83
hc_plot_longitudinal_cap	86
hc_plot_longitudinal_endotype_meta_flow	87
hc_plot_longitudinal_endotypes	88
hc_plot_longitudinal_enrichment_meta_waves	89
hc_plot_longitudinal_enrichment_waves	91
hc_plot_longitudinal_k_criterion	94
hc_plot_longitudinal_meta_consensus	95
hc_plot_longitudinal_meta_embeddings	97
hc_plot_longitudinal_meta_module_waves	98
hc_plot_longitudinal_module_clusters	100
hc_plot_longitudinal_module_means	102
hc_plot_sample_distributions	103
hc_preview_celltype_database	104
hc_print_longitudinal_endotypes	105
hc_read_data	106
hc_read_supplementary	107
hc_regroup_samples	108
hc_resolve_paths	110
hc_run_expression_analysis_1	112
hc_run_expression_analysis_2	113
hc_satellite	114
hc_set_cutoff	115
hc_set_global_settings	116
hc_set_layer_settings	118
hc_set_paths	120
hc_set_supp_files	121
hc_split_modules	122
hc_suggest_topvar	124
hc_tf_overrep_module	124
hc_tf_overrep_network	125
hc_tune_cutoff	126
hc_tune_resolution	128
hc_unsplit_modules	129
hc_update_clustering_algorithm	130
hc_upstream_inference	130
hc_user_specific_cluster_profiling	134
hc_visualize_gene_expression	135
hc_write_session_info	136
HCoCenaClasses	137
plot.hc_sample_regrouping	137

 hc_add_meta_time_grouping

Add meta-cluster x time grouping column for heatmap regrouping

Description

Creates an annotation column that combines donor meta-cluster assignment and timepoint labels (e.g. MC1__24h). This column can be used as grouping_v in hc_run_expression_analysis_2() to build a cluster heatmap by meta-cluster x timepoint.

Usage

```
hc_add_meta_time_grouping(
  hc,
  donor_col = "Subject",
  time_col = "Time_token",
  grouping_col = "meta_cluster_time",
  slot_name = "longitudinal_endotypes",
  layer = "all",
  time_levels = NULL,
  meta_cluster_levels = NULL,
  separator = "__",
  append_layer_suffix = FALSE
)
```

Arguments

hc	A HCoCenaExperiment.
donor_col	Annotation column containing donor IDs.
time_col	Annotation column containing timepoint labels.
grouping_col	Name of the new annotation column to create.
slot_name	Satellite slot holding longitudinal outputs. Must contain meta_cluster table. Use a concrete slot name, a shared prefix such as "longitudinal_endotypes" to process matching suffixed slots, or "all" to use every compatible step 2 slot in hc@satellite.
layer	Layer index, layer id/name, or "all" (default) to add the grouping column to all layers.
time_levels	Optional explicit timepoint order vector.
meta_cluster_levels	Optional explicit meta-cluster order vector.
separator	String between meta-cluster and timepoint labels.
append_layer_suffix	Logical. If TRUE, append layer id to each group label (recommended when multiple layers share group names).

Value

Updated HCoCenaExperiment. The computed order is stored in `hc@satellite[[slot_name]]$meta_time_grouping` and/or the resolved per-layer step 2 slots.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc
hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
  method = "kmeans",
  consensus = TRUE,
  seed = 1
)
hc <- hc_add_meta_time_grouping(hc, donor_col = "donor", time_col = "timepoint")
```

hc_algo_alluvial	<i>algo alluvial</i>
------------------	----------------------

Description

Alluvial comparison plots (S4 API)

Usage

```
hc_algo_alluvial(hc)
```

Arguments

hc A HCoCenaExperiment.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_algo_alluvial(hc)
```

hc_antilog	<i>Reverse a log transformation</i>
------------	-------------------------------------

Description

Inverse of `base::log()` for an arbitrary base, i.e. `hc_antilog(x, 2)` returns 2^x .

Usage

```
hc_antilog(lx, base)
```

Arguments

lx	Numeric vector, matrix or data frame of logged values.
base	The base the logarithm was taken to.

Value

The de-logged values, in the same shape as `lx`.

Examples

```
hc_antilog(c(1, 2, 3), base = 2)
```

hc_auto_set_paths	<i>Resolve and apply standard paths in one call</i>
-------------------	---

Description

Resolve and apply standard paths in one call

Usage

```
hc_auto_set_paths(
  hc,
  project_root_hint = NULL,
  local_count_subdir = "count_data",
  local_annotation_subdir = "annotation_data",
  local_reference_subdir = "reference_files",
  local_output_subdir = "output",
  docker_reference_dir = "/home/rstudio/reference_files",
```

```

docker_count_dir = "/home/rstudio/count_data",
docker_annotation_dir = "/home/rstudio/annotation_data",
docker_output_dir = "/home/rstudio/output",
create_docker_dirs = TRUE,
fallback_count_data = "PATH_TO_FOLDER_THAT_HOLDS_YOUR_GENE_EXPRESSION_TABLES/",
fallback_annotation = "PATH_TO_FOLDER_THAT_HOLDS_YOUR_ANNOTATION_TABLES/",
fallback_reference_files = "PATH_TO_REFERENCE_FILES_FOLDER/",
fallback_output = "PATH_TO_FOLDER_WHERE_HCOECENA_SHOULD_SAVE_ALL_ANALYSIS_OUTPUTS/",
dir_count_data = NULL,
dir_annotation = NULL,
dir_reference_files = NULL,
dir_output = NULL
)

```

Arguments

hc	A HCoCenaExperiment.
project_root_hint	Optional local project root to prioritize if it exists. If missing or not found, the current working directory is used.
local_count_subdir	Local count-data subfolder below project_root_hint.
local_annotation_subdir	Local annotation-data subfolder below project_root_hint.
local_reference_subdir	Local reference-files subfolder below project_root_hint.
local_output_subdir	Local output subfolder below project_root_hint.
docker_reference_dir	Docker reference-files directory used to detect Docker mode.
docker_count_dir	Docker count-data directory.
docker_annotation_dir	Docker annotation-data directory.
docker_output_dir	Docker output directory.
create_docker_dirs	Boolean. If TRUE and Docker mode is detected, create the Docker directories when missing.
fallback_count_data	Fallback count-data directory string used outside Docker when no local folder was detected.
fallback_annotation	Fallback annotation directory string used outside Docker when no local folder was detected.

fallback_reference_files Fallback reference-files directory string used outside Docker when no local folder was detected.
fallback_output Fallback output directory string used outside Docker when no local folder was detected.
dir_count_data Optional explicit override for count-data directory.
dir_annotation Optional explicit override for annotation directory.
dir_reference_files Optional explicit override for reference-files directory.
dir_output Optional explicit override for output directory.

Value

Updated HCoCenaExperiment.

Examples

```

root <- file.path(tempdir(), "hcocena-auto-paths")
dir.create(file.path(root, "reference_files"), recursive = TRUE, showWarnings = FALSE)
dir.create(file.path(root, "output"), recursive = TRUE, showWarnings = FALSE)
hc <- hc_init()
hc <- hc_auto_set_paths(
  hc,
  project_root_hint = root,
  local_count_subdir = NULL,
  local_annotation_subdir = NULL
)
as.data.frame(methods::slot(hc_config(hc), "paths"))
  
```

hc_auto_tune

Automatically tune key hCoCena parameters

Description

hc_auto_tune() recommends and optionally applies:

1. Layer-wise cutoffs based on run_expression_analysis_1() output.
2. Clustering settings based on modularity across candidate algorithms.

Usage

```

hc_auto_tune(
  hc,
  tune_cutoff = TRUE,
  tune_clustering = TRUE,
  apply = TRUE,

```

```

cluster_algorithms = c("auto", "cluster_leiden", "cluster_louvain"),
resolution_grid = base::seq(0.05, 0.5, by = 0.05),
no_of_iterations = 2,
partition_type = "RBConfigurationVertexPartition",
max_cluster_count_per_gene = 1,
module_count_bounds = c(3, 60),
prefer_resolution = 0.1,
verbose = TRUE
)

```

Arguments

hc	A HCoCenaExperiment object.
tune_cutoff	Logical; tune per-layer correlation cutoffs.
tune_clustering	Logical; tune clustering algorithm / resolution.
apply	Logical; if TRUE, apply best recommendations to hc.
cluster_algorithms	Character vector of candidate algorithms for clustering tuning. Supported: "auto", "cluster_leiden", "cluster_louvain", "cluster_fast_greedy", "cluster_walktrap", "cluster_infomap", "cluster_label_prop".
resolution_grid	Numeric vector of Leiden resolutions to evaluate for "auto" and "cluster_leiden".
no_of_iterations	Passed to hc_cluster_calculation().
partition_type	Passed to hc_cluster_calculation(). Note: for Leiden with "ModularityVertexPartition", sweeping resolution_grid usually does not change the partition. If you want resolution-driven module-count tuning, use "CPMVertexPartition" or "RBConfigurationVertexPartition".
max_cluster_count_per_gene	Passed to hc_cluster_calculation().
module_count_bounds	Optional length-2 numeric vector (min_modules, max_modules) used as a soft constraint during ranking.
prefer_resolution	Tie-break preference for Leiden resolution.
verbose	Logical; print progress messages.

Details

The tuning report is stored in `hc@satellite$auto_tune`.

Value

Updated HCoCenaExperiment with tuning report in `hc@satellite$auto_tune`.

Examples

```
hc <- hc_example_data("after_part1")
hc <- hc_auto_tune(hc, apply = FALSE)
```

```
hc_auto_tune_cutoff_tiered
```

Tiered and robust cutoff recommendation

Description

Alternative cutoff tuning method that can be compared against the current simple cutoff recommendation.

Usage

```
hc_auto_tune_cutoff_tiered(
  hc,
  apply = FALSE,
  tier1_sft_min = 0.85,
  tier1_mean_k_min = 10,
  tier2_sft_min = 0.8,
  tier2_mean_k_min = 5,
  fallback_sft_min = 0.7,
  require_negative_slope = TRUE,
  max_no_networks = NULL,
  max_components = NULL,
  min_node_fraction = NULL,
  fallback_respect_base_filters = TRUE,
  verbose = TRUE
)
```

Arguments

hc	A HCoCenaExperiment object.
apply	Logical; if TRUE, apply the resulting cutoff vector with <code>hc_set_cutoff()</code> . Missing tiered recommendations fall back to existing cutoffs from <code>hc@config@layer\$cutoff</code> where possible.
tier1_sft_min	Numeric minimum SFT fit (R.squared) for the strict rule.
tier1_mean_k_min	Numeric minimum mean connectivity (mean_k) for the strict rule.
tier2_sft_min	Numeric minimum SFT fit (R.squared) for the relaxed rule.
tier2_mean_k_min	Numeric minimum mean connectivity (mean_k) for the relaxed rule.
fallback_sft_min	Numeric lower bound for best_available selection using the maximum available SFT fit.

`require_negative_slope` Logical; require a negative slope ($\text{slope} < 0$) in Tier 1/Tier 2 (and optionally fallback, see `fallback_respect_base_filters`).

`max_no_networks` Optional numeric scalar. If set, only cutoffs with `no_of_networks` $\leq$ `max_no_networks` are accepted in Tier 1/Tier 2.

`max_components` Deprecated alias for `max_no_networks` (kept for backward compatibility).

`min_node_fraction` Optional numeric in $(0, 1]$. If set, require `no_nodes` $\geq$ `min_node_fraction` * `max(no_nodes)` in Tier 1/Tier 2.

`fallback_respect_base_filters` Logical; if TRUE, best_available selection is restricted by base filters (`slope`, `max_no_networks`, `min_node_fraction`).

`verbose` Logical; print per-layer recommendation messages.

Details

For each layer, the function:

1. Summarizes cutoff statistics.
2. Computes additional diagnostics (`slope`, `mean_k`).
3. Applies a staged rule set (`strict`, `relaxed`, `best_available`).

Results are stored in `hc@satellite$cutoff_tuning` (and mirrored to `hc@satellite$auto_tune_cutoff_tiered` for compatibility).

Value

Updated `HCoCenaExperiment` with a report in `hc@satellite$cutoff_tuning`.

Examples

```
hc <- hc_example_data("after_part1")
hc <- hc_auto_tune_cutoff_tiered(hc, apply = FALSE, verbose = FALSE)
```

`hc_build_integrated_network`
Network integration

Description

The previously constructed layer-specific networks are being integrated.

Usage

```
hc_build_integrated_network(
  hc,
  mode = "u",
  with = NULL,
  multi_edges = "min",
  GFC_when_missing = NULL
)
```

Arguments

hc	A HCoCenaExperiment.
mode	A string, either "u", if network integration is to be done by union (default), or "i", if integration is to be done by intersection. For details please refer to the information pages provided in the repository's Wiki.
with	Either an integer giving the number of the dataset to be used as reference (e.g., 1) or the name given to the layer. Can be ignored when integration is done by union.
multi_edges	One of "min", "mean" or "max" resulting in the simplification of the multigraph by using the minimum, the mean or the maximum edge weight among the multiple edges, respectively. Multiple edges occur when the edge was present in more than one dataset.
GFC_when_missing	The value to substitute missing data in the case where some genes were not measured in all but only some of the datasets.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("after_part2")
hc <- hc_build_integrated_network(hc, mode = "u")
```

hc_celltype_activity_decoupler

Module cell-type activity from Enrichr markers via decoupleR (S4 API)

Description

Module cell-type activity from Enrichr markers via decoupleR (S4 API)

Usage

```

hc_celltype_activity_decoupler(
  hc,
  databases = c("Descartes_Cell_Types_and_Tissue_2021", "Human_Gene_Atlas"),
  custom_gmt_files = NULL,
  clusters = c("all"),
  mode = c("coarse", "fine"),
  top = 3,
  qval = 0.1,
  padj = "BH",
  activity_input = c("gfc", "fc", "expression"),
  fc_comparisons = NULL,
  method = "ulm",
  minsize = 5,
  min_term_genes = 5,
  annotation_slot = c("auto", "enriched_per_cluster", "enriched_per_cluster2"),
  slot_suffix = "decoupler",
  clear_previous_slots = FALSE,
  coarse_map = NULL,
  coarse_include_other = TRUE,
  refresh_db = FALSE,
  export_excel = TRUE,
  excel_file = "Module_Celltype_Activity_decoupler.xlsx",
  plot_heatmap = FALSE,
  heatmap_file_name = "Heatmap_modules_celltype_activity_decoupler.pdf",
  ...
)

```

Arguments

hc	A HCoCenaExperiment.
databases	Character vector of Enrichr library names.
custom_gmt_files	Optional custom GMT file(s) to include as additional marker resources. Accepts a character vector or named list of file paths.
clusters	Either "all" (default) or a character vector of module IDs/colors.
mode	Either "coarse" (default) or "fine".
top	Number of selected marker activities per module.
qval	Adjusted p-value cutoff.
padj	Multiple-testing correction method.
activity_input	One of "gfc" (default), "fc", "expression".
fc_comparisons	Vector like c("A_vs_B", "C_vs_B") when activity_input = "fc".
method	decoupleR method suffix (e.g. "ulm").
minsize	Minimum target size passed to decoupleR.
min_term_genes	Minimum genes per Enrichr term.

annotation_slot	Legacy slot override for single-database runs.
slot_suffix	Optional slot suffix; default "decoupler".
clear_previous_slots	If TRUE, clears old enriched_per_cluster* slots first.
coarse_map	Optional named regex map used in coarse mode.
coarse_include_other	Keep "Other" class in coarse mode.
refresh_db	Refresh Enrichr cache.
export_excel	Write summary workbook.
excel_file	Excel filename.
plot_heatmap	Replot cluster heatmap with dynamic slots.
heatmap_file_name	Heatmap filename when plot_heatmap = TRUE.
...	Further arguments passed through to the underlying implementation. Every supported argument is documented on this page; ... only carries the tail of the argument list.

Value

Updated HCoCenaExperiment.

Examples

```
if (requireNamespace("decoupleR", quietly = TRUE)) {
  hc <- hc_example_data("clustered")
  gmt <- system.file("extdata", "toy_celltype_markers.gmt",
    package = "hcocena")
  hc <- hc_celltype_activity_decoupler(
    hc,
    databases = character(),
    custom_gmt_files = c(CellTypes = gmt),
    export_excel = FALSE
  )
  names(hc_satellite(hc, "celltype_activity_decoupler"))
}
```

hc_celltype_annotation

Module cell-type annotation from Enrichr (S4 API)

Description

Module cell-type annotation from Enrichr (S4 API)

Usage

```

hc_celltype_annotation(
  hc,
  databases = c("Descartes_Cell_Types_and_Tissue_2021", "Human_Gene_Atlas"),
  custom_gmt_files = NULL,
  clusters = c("all"),
  mode = c("coarse", "fine"),
  top = 3,
  qval = 0.1,
  padj = "BH",
  min_term_genes = 5,
  min_gs_size = 10,
  max_gs_size = 5000,
  annotation_slot = c("auto", "enriched_per_cluster", "enriched_per_cluster2"),
  slot_suffix = NULL,
  clear_previous_slots = TRUE,
  coarse_map = NULL,
  coarse_include_other = TRUE,
  refresh_db = FALSE,
  export_excel = TRUE,
  excel_file = "Module_Celltype_Annotation.xlsx",
  plot_heatmap = FALSE,
  heatmap_file_name = "Heatmap_modules_celltype_annotation.pdf",
  ...
)

```

Arguments

hc	A HCoCenaExperiment.
databases	Character vector of Enrichr library names.
custom_gmt_files	Optional custom GMT file(s) to include in the cell-type annotation. Accepts a character vector or named list of file paths. Paths can be absolute/relative or file names inside <code>dir_reference_files</code> .
clusters	Either "all" (default) or a character vector of module IDs/colors.
mode	Either "coarse" (broad classes) or "fine" (specific terms).
top	Number of selected categories per module.
qval	Maximum adjusted p-value (qvalue) for significant terms.
padj	Multiple-testing correction method for <code>clusterProfiler::enricher()</code> .
min_term_genes	Minimum number of genes required per Enrichr term.
min_gs_size	Minimum gene-set size used in <code>clusterProfiler::enricher()</code> (<code>minGSSize</code>).
max_gs_size	Maximum gene-set size used in <code>clusterProfiler::enricher()</code> (<code>maxGSSize</code>).
annotation_slot	Backward-compatibility option when only one database is used. With multiple databases, one slot per DB is always written (<code>enriched_per_cluster_<db></code>), and previous DB slots are reset on each run.

slot_suffix	Optional character suffix appended to generated annotation slots (for example "decoupler" -> enriched_per_cluster_<db>_decoupler). Useful to keep multiple annotation runs side by side.
clear_previous_slots	Logical. If TRUE (default), previous enriched_per_cluster* slots are removed before writing new results. Set to FALSE to keep existing slots.
coarse_map	Optional named character vector with regex rules for mode = "coarse". Names are output class labels.
coarse_include_other	Logical. Keep unmatched terms as "Other" in mode = "coarse".
refresh_db	Logical. If TRUE, re-download Enrichr metadata/libraries.
export_excel	Logical. If TRUE, write summary tables to Excel.
excel_file	Excel file name in the configured output folder.
plot_heatmap	Logical. If TRUE, run hc_plot_cluster_heatmap() after updating annotation slots.
heatmap_file_name	File name used when plot_heatmap = TRUE.
...	Further arguments passed through to the underlying implementation. Every supported argument is documented on this page; ... only carries the tail of the argument list.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
gmt <- system.file("extdata", "toy_celltype_markers.gmt", package = "hcocena")
# A local marker GMT instead of the default Enrichr libraries, so that the
# example runs offline:
hc <- hc_celltype_annotation(
  hc,
  databases = character(),
  custom_gmt_files = c(CellTypes = gmt),
  export_excel = FALSE
)
hc_satellite(hc, "celltype_annotation")$selected_celltypes
```

hc_change_grouping_parameter
change grouping parameter

Description

Recalculate grouped GFCs and replot module heatmap (S4 API)

Usage

```
hc_change_grouping_parameter(
  hc,
  group_by,
  col_order = NULL,
  cluster_columns = FALSE,
  row_order = NULL,
  cluster_rows = TRUE
)
```

Arguments

hc	A HCoCenaExperiment.
group_by	Grouping column to use (must be present in all annotation tables).
col_order	Optional heatmap column order.
cluster_columns	Whether to cluster heatmap columns. Default is FALSE, so the stored main hCoCena column order is reused when available.
row_order	Optional heatmap row order.
cluster_rows	Whether to cluster heatmap rows.

Details

Uses the existing module definitions from the current object and only recalculates grouped GFCs for a different grouping variable. No integrated network rebuild and no module re-clustering is performed.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_change_grouping_parameter(hc, group_by = "batch")
```

hc_check_dirs

Fixes Directories

Description

Iteratively calls fix_dir() on the provided working directory paths to fix them if necessary or throw an error if they are invalid.

Usage

```
hc_check_dirs(hc, create_output_dir = TRUE)
```

Arguments

hc A HCoCenaExperiment.
 create_output_dir Boolean. If TRUE and dir_output is missing, create it.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("prepared")
hc <- hc_set_paths(
  hc,
  dir_count_data = FALSE,
  dir_annotation = FALSE,
  dir_reference_files = FALSE,
  dir_output = tempdir()
)
hc <- hc_check_dirs(hc)
```

hc_check_tf	<i>Check TF targets (S4 API)</i>
-------------	----------------------------------

Description

Check TF targets (S4 API)

Usage

```
hc_check_tf(hc, TF)
```

Arguments

hc A HCoCenaExperiment.
 TF Transcription factor symbol.

Value

Updated HCoCenaExperiment.

Examples

```
# Needs the ChEA3 results of hc_tf_overrep_network(), which queries the
# ChEA3 web service.
hc <- hc_example_data("clustered")
hc <- hc_tf_overrep_network(hc, topTF = 10, topTarget = 5)
tfs <- names(hc_satellite(hc, "tf_network_targets"))
if (length(tfs) > 0) {
  hc <- hc_check_tf(hc, TF = tfs[[1]])
}
```

hc_cluster_calculation

Cluster Calculation

Description

The function offers several community detection algorithms to identify dense regions in the co-expression network. These dense regions represent collections of highly co-expressed genes likely to form a functional group.

Usage

```
hc_cluster_calculation(
  hc,
  cluster_algo = "cluster_leiden",
  no_of_iterations = 2,
  resolution = 0.1,
  partition_type = "RBConfigurationVertexPartition",
  max_cluster_count_per_gene = 1,
  return_result = FALSE
)
```

Arguments

hc	A HCoCenaExperiment.
cluster_algo	The clustering algorithm to be used. The choice is between "cluster_leiden" (default), "cluster_louvain", "cluster_label_prop", "cluster_fast_greedy", "cluster_infomap", "cluster_walktrap" and "auto" (in which case all are tested and the one with the highest modularity is chosen).
no_of_iterations	Some of the algorithms are iterative (e.g. Leiden Algorithm). Set here, how many iterations should be performed. For information on which other algorithms are iterative, please refer to their documentation in the igraph or leiden-base package. Default is 2.
resolution	The cluster resolution if the cluster algorithm is set to "cluster_leiden". Default is 0.1. Higher values result in more clusters and vice versa.

partition_type Name of the partition type. Select from 'CPMVertexPartition', 'ModularityVertexPartition', 'RBConfigurationVertexPartition' and 'RBERVertexPartition'. Default is 'RBConfigurationVertexPartition'.

max_cluster_count_per_gene The maximum number of different clusters a gene is allowed to be associated with during the different iterations before it is marked as indecisive and removed. Default is 1.

return_result Logical. If TRUE, return the cluster table instead of storing it in hcobject.

Value

Updated HCoCenaExperiment. If `return_result = TRUE` and the clustering backend returns a cluster table, that table is returned instead.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_cluster_calculation(hc, cluster_algo = "cluster_leiden")
```

hc_clusters	<i>Access cluster information from an HCoCenaExperiment</i>
-------------	---

Description

Access cluster information from an HCoCenaExperiment

Usage

```
hc_clusters(x)

## S4 method for signature 'HCoCenaExperiment'
hc_clusters(x)
```

Arguments

x An HCoCenaExperiment.

Value

Cluster information or NULL if missing.

Examples

```
hc <- hc_init()
hc_clusters(hc)
```

hc_col_anno_categorical

Add categorical module heatmap annotations (S4 API)

Description

Add categorical module heatmap annotations (S4 API)

Usage

```
hc_col_anno_categorical(hc, variables, variable_label = NULL, type = "abs")
```

Arguments

hc	A HCoCenaExperiment.
variables	Character vector of categorical annotation column names to add as heatmap column annotations.
variable_label	Optional display label(s) for variables. If NULL (default), the column names themselves are used.
type	Either "abs" (default) for absolute counts or "rel" for relative proportions.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_col_anno_categorical(hc, variables = "batch")
```

hc_col_anno_numerical

Add numeric module heatmap annotations (S4 API)

Description

Adds numeric annotation tracks to the columns of the module heatmap. Call with `variables = NULL` to remove previously added tracks.

Usage

```
hc_col_anno_numerical(hc, variables, variable_label)
```

Arguments

`hc` A HCoCenaExperiment.
`variables` Character vector of numeric annotation column names to add. Use NULL to remove existing numeric annotations.
`variable_label` Display label for the annotation track.

Value

Updated HCoCenaExperiment.

Examples

```

hc <- hc_example_data("clustered")
hc <- hc_col_anno_numerical(hc, variables = "batch", variable_label = "batch")

```

hc_colour_single_cluster

Highlight single cluster in network (S4 API)

Description

Re-plots the integrated network with one module emphasised by colour and node size.

Usage

```
hc_colour_single_cluster(hc, cluster)
```

Arguments

`hc` A HCoCenaExperiment.
`cluster` The module to highlight, given as its colour or module label.

Value

Updated HCoCenaExperiment.

Examples

```

hc <- hc_example_data("clustered")
hc <- hc_colour_single_cluster(hc, cluster = "gold")

```

hc_config	<i>Access configuration from an HCoCenaExperiment</i>
-----------	---

Description

Access configuration from an HCoCenaExperiment

Usage

```
hc_config(x)

## S4 method for signature 'HCoCenaExperiment'
hc_config(x)
```

Arguments

x An HCoCenaExperiment.

Value

A HCoCenaConfig object.

Examples

```
hc <- hc_init()
class(hc_config(hc))
```

hc_cut_hclust	<i>Cut a hierarchical clustering tree to regroup samples (S4 API)</i>
---------------	---

Description

Assigns new sample group labels from the data structure rather than from metadata, by cutting a hierarchical clustering tree. Useful when the variable of interest does not match the structure seen in the heatmaps or the PCA.

Usage

```
hc_cut_hclust(
  hc,
  by = "all",
  set = "all",
  method = "complete",
  k,
  save = FALSE
)
```

Arguments

hc	A HCoCenaExperiment.
by	What to regroup on: "all" (default, all genes; dendrogram only, for memory reasons), "network" (network genes) or "module" (mean expression per module). The latter two also draw a heatmap.
set	Layers to regroup: "all" (default) or an integer vector of layer indices.
method	Agglomeration method passed to <code>stats::hclust()</code> . Default is "complete".
k	Integer vector giving, per regrouped layer, the number of clusters to cut the dendrogram into. Defaults to 1 per layer (no cutting - good for a first look).
save	Logical. If TRUE, overwrite the sample groups and export a new annotation file with the <code>_regrouped</code> suffix; the original groups are kept in a <code><voi>_old</code> column. Default is FALSE.

Details

Re-run with `save = FALSE` to try different `k` values; only set `save = TRUE` once you have settled on one, since that overwrites the original groups and exports a new annotation file (without row names). After saving, re-run the pipeline from `hc_run_expression_analysis_2()` onward.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_cut_hclust(hc, by = "module", k = c(2, 2), save = FALSE)
```

hc_define_layers	<i>Define layers in an HCoCenaExperiment</i>
------------------	--

Description

Define layers in an HCoCenaExperiment

Usage

```
hc_define_layers(hc, data_sets = list())
```

Arguments

hc	A HCoCenaExperiment.
data_sets	Named list of layers, each value a pair (count, annotation).

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_init()
hc <- hc_define_layers(
  hc,
  data_sets = list(
    Layer1 = c("counts.tsv", "anno.tsv")
  )
)
as.data.frame(methods::slot(hc_config(hc), "layer"))
```

hc_example_data

Load an example HCoCenaExperiment

Description

Returns one of the small example objects shipped in `inst/extdata`, taken at a given stage of the standard workflow, so that individual steps can be tried without running everything before them. The data are two simulated layers of 40 marker genes for four blood cell types over 32 samples; see `inst/scripts/make-fixtures.R` for the generating code.

Usage

```
hc_example_data(
  stage = c("prepared", "after_part1", "after_part2", "clustered"),
  dir_output = NULL
)
```

Arguments

stage	Workflow stage to load: • "prepared": data read and all settings made; • "after_part1": correlations computed and the cutoff chosen (<code>hc_run_expression_analysis_1()</code> <code>hc_set_cutoff()</code>); • "after_part2": group fold changes computed (<code>hc_run_expression_analysis_2()</code>); • "clustered": integrated network built and modules detected (<code>hc_build_integrated_network()</code>, <code>hc_cluster_calculation()</code>).
dir_output	Output directory to store in the object. Defaults to a new directory inside <code>tempdir()</code> .

Details

The output directory stored in the object is replaced by a fresh directory inside `tempdir()`, so functions that write plots or tables never write outside the session's temporary directory.

Value

A HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc
table(hc_gene_to_cluster(hc)$color)
```

hc_export_clusters	<i>export clusters</i>
--------------------	------------------------

Description

Export clusters (S4 API)

Usage

```
hc_export_clusters(hc)
```

Arguments

hc A HCoCenaExperiment.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_export_clusters(hc)
```

hc_export_to_cytoscape	<i>Export network to Cytoscape (S4 API)</i>
------------------------	---

Description

Export network to Cytoscape (S4 API)

Usage

```
hc_export_to_cytoscape(hc, name = "my igraph", docker_container = FALSE)
```

Arguments

hc	A HCoCenaExperiment.
name	Name under which the network is created in Cytoscape. Default is "my igraph".
docker_container	Logical. Set to TRUE when running hCoCena inside a Docker container that talks to a Cytoscape instance on the host. Default is FALSE.

Value

Updated HCoCenaExperiment.

Examples

```
# Needs a running Cytoscape instance reachable through RCy3.
cytoscape_up <- requireNamespace("RCy3", quietly = TRUE) &&
  isTRUE(tryCatch({
    RCy3::cytoscapePing()
    TRUE
  }, error = function(e) FALSE))
if (cytoscape_up) {
  hc <- hc_example_data("clustered")
  hc <- hc_export_to_cytoscape(hc, name = "hcocena example")
}
```

hc_export_to_local_folder

Export network to local folder (S4 API)

Description

Export network to local folder (S4 API)

Usage

```
hc_export_to_local_folder(hc, file)
```

Arguments

hc	A HCoCenaExperiment.
file	Target folder for the exported network files. If omitted, the current output/save folder configured in hc is used.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_export_to_local_folder(hc, file = tempdir())
```

hc_find_hubs	<i>Hub detection (S4 API)</i>
--------------	-------------------------------

Description

Determines hub genes per module from a combined ranking of weighted degree, closeness and betweenness centrality.

Usage

```
hc_find_hubs(
  hc,
  clusters = c("all"),
  top = 10,
  tree_layout = FALSE,
  TF_only = FALSE,
  save = FALSE,
  plot = FALSE
)
```

Arguments

hc	A HCoCenaExperiment.
clusters	Either "all" (default) or a vector of module colours for which hub detection should be performed.
top	Integer. Number of top-ranked genes per module to report as hubs. Default is 10.
tree_layout	Logical. Draw the per-module hub network with a tree layout. Default is FALSE.
TF_only	Either FALSE (default, consider all genes), "all" (only genes from the transcription-factor reference file), or one gene category from that file's last column.
save	Logical. Write the hub network and expression heatmap to PDF. Default is FALSE.
plot	Logical. Draw the per-module network. Default is FALSE.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_find_hubs(hc)
```

`hc_functional_enrichment`*Functional enrichment (S4 API)*

Description

Functional enrichment (S4 API)

Usage

```
hc_functional_enrichment(  
  hc,  
  gene_sets = c("Go", "Kegg", "Hallmark", "Reactome"),  
  custom_gmt_files = NULL,  
  top = 5,  
  clusters = c("all"),  
  padj = "BH",  
  qval = 0.05,  
  universe = c("all_genes", "network"),  
  consistent_terms = TRUE,  
  heatmap_side = "left",  
  heatmap_cluster_rows = FALSE,  
  cluster_columns = FALSE,  
  heatmap_cluster_columns = NULL,  
  heatmap_show_row_dend = FALSE,  
  heatmap_show_column_dend = FALSE,  
  col_order = NULL,  
  heatmap_col_order = NULL,  
  heatmap_order = NULL,  
  heatmap_module_label_mode = "same",  
  heatmap_show_gene_counts = FALSE,  
  heatmap_column_label_fontsize = NULL,  
  heatmap_module_label_fontsize = NULL,  
  legend_fontsize = NULL,  
  enrichment_label_fontsize = NULL,  
  enrichment_db_header_fontsize = NULL,  
  enrichment_label_wrap = FALSE,  
  enrichment_label_wrap_width = 30,  
  gfc_scale_limits = NULL,  
  store_panel_objects = c("auto", "always", "never"),  
  pdf_width = NULL,  
  pdf_height = NULL,  
  pdf_pointsize = 11,  
  ...  
)
```

Arguments

hc	A HCoCenaExperiment.
gene_sets	A vector. The names of databases enrichment should be performed for. Choose one or multiple of "Go", "Kegg", "Hallmark", and/or "Reactome". Available databases depend on supplement files previously set Default is "Hallmark".
custom_gmt_files	Optional custom GMT file(s) to include directly in enrichment. Accepts a character vector or named list of file paths. Unnamed entries are auto-labeled (CustomEnrichment1, ...). Paths can be absolute/relative or file names inside dir_reference_files.
top	Integer. The number of most strongly enriched terms to return per cluster. Default is 5.
clusters	Either "all" (default) or a vector of clusters as strings. Defines for which clusters to perform the enrichment.
padj	Method to use for multiple testing correction. Can be one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". Default is "BH" (Benjamini-Hochberg). The reported qvalue column holds the p-values adjusted with this method, and qval is applied to it. (Up to and including 0.99.7 the qvalue column carried clusterProfiler's Storey q-value, which is computed independently of padj, so this argument did not affect which terms were called significant. Result tables from earlier versions can therefore differ.)
qval	Upper threshold for the adjusted p-value. Default is 0.05.
universe	Background gene set for the hypergeometric test. "all_genes" (default) uses every gene measured in any layer, asking whether a module is enriched relative to the transcriptome. "network" uses only the genes that entered the integrated network – the genes that could have landed in a module at all – asking whether a module is enriched relative to the other modules. "network" removes the bias that the top-variance selection and the correlation cutoff introduce, and typically returns fewer but more module-specific terms.
consistent_terms	Logical. If TRUE (default), use the union of the top top enriched terms across the selected modules for each database and also show those terms in other modules whenever they are significantly enriched there. If FALSE, show the top enriched terms per module separately.
heatmap_side	Position of the hCoCena heatmap in the combined output. Choose one of "left" (default) or "right".
heatmap_cluster_rows	A Boolean whether or not to cluster rows in the hCoCena heatmap.
cluster_columns	A Boolean whether or not to cluster columns in the hCoCena heatmap.
heatmap_cluster_columns	Legacy alias for cluster_columns.
heatmap_show_row_dend	A Boolean whether to show the row dendrogram when heatmap_cluster_rows = TRUE.

heatmap_show_column_dend	A Boolean whether to show the column dendrogram when <code>cluster_columns = TRUE</code> .
col_order	Optional character vector overriding the hCoCena heatmap column order for this enrichment plot only. If NULL (default), the column order from the main module heatmap is reused when available.
heatmap_col_order	Legacy alias for <code>col_order</code> .
heatmap_order	Optional character vector specifying module order in the hCoCena heatmap. Entries can be module colors (e.g. "turquoise") or module labels from the main heatmap (e.g. "M1", "M2", ...). Modules not listed are appended afterwards.
heatmap_module_label_mode	Controls labels inside module color boxes of the hCoCena heatmap. One of "same" or "none". "same" reuses the prefix from <code>hc_plot_cluster_heatmap()</code> and reindexes modules consecutively (M1, M2, ...) after final module ordering.
heatmap_show_gene_counts	A Boolean whether or not to show gene counts per module in the right annotation of the hCoCena heatmap. Default is FALSE.
heatmap_column_label_fontsize	Optional numeric font size for hCoCena heatmap column labels in enrichment plots. If NULL (default), uses automatic sizing.
heatmap_module_label_fontsize	Optional numeric font size for module labels inside module color boxes (M1, M2, ...). If NULL (default), uses automatic sizing.
legend_fontsize	Optional numeric base font size for enrichment-related legends (GFC legend and enrichment significance legend). If NULL (default), uses automatic sizing.
enrichment_label_fontsize	Optional numeric font size for enrichment term labels. If NULL (default), uses automatic sizing.
enrichment_db_header_fontsize	Optional numeric font size for database headers in combined all-DB enrichment plots (e.g. "Go", "Kegg", "Hallmark"). If NULL (default), uses automatic sizing.
enrichment_label_wrap	Logical. If TRUE, wraps enrichment term labels using <code>enrichment_label_wrap_width</code> . Default is FALSE.
enrichment_label_wrap_width	Integer wrap width used when <code>enrichment_label_wrap = TRUE</code> . Default is 30.
gfc_scale_limits	Optional numeric vector controlling the module-heatmap color scale limits used in enrichment plots. Provide one positive number ($x \rightarrow c(-x, x)$) or two numbers ($c(\min, \max)$). If NULL, uses stored limits from the latest main module heatmap when available, otherwise falls back to $c(-\text{range_GFC}, \text{range_GFC})$.
store_panel_objects	One of "auto" (default), "always", or "never". Controls whether heavy heatmap/panel objects are stored inside hc for later redraw with <code>hc_plot_enrichment_panels()</code> .

	"auto" stores them only for a single selected database; multi-database runs keep only tables to save memory.
pdf_width	Optional numeric width (inches) for enrichment PDFs. If NULL (default), width is auto-estimated from content.
pdf_height	Optional numeric height (inches) for enrichment PDFs. If NULL (default), height is auto-estimated from content.
pdf_pointsize	Numeric base pointsize used for PDF devices. Default is 11.
...	Further arguments passed through to the underlying implementation. Every supported argument is documented on this page; ... only carries the tail of the argument list.

Value

Updated HCoCenaExperiment.

Significance columns

The result tables carry pvalue (raw), p_adjusted (corrected with the method given in padj, and what the qval filter and the ranking use), padj_method (which correction that was) and q_storey (clusterProfiler's Storey q-value, NA where pi0 could not be fitted - which is common for small gene-set collections). qvalue is kept as an alias of p_adjusted so existing scripts and exports keep working.

Examples

```
hc <- hc_example_data("clustered")
gmt <- tempfile(fileext = ".gmt")
writeLines(
  c(
    paste(c("T cells", "-", "CD3D", "CD3E", "CD3G", "CD2", "CD28",
            "LCK", "ZAP70", "IL7R", "CD7", "TRAC"), collapse = "\t"),
    paste(c("B cells", "-", "CD19", "MS4A1", "CD79A", "CD79B", "BLNK",
            "PAX5", "CR2", "FCRL1", "TNFRSF13B", "VPREB3"), collapse = "\t"),
    paste(c("Monocytes", "-", "CD14", "LYZ", "FCN1", "VCAN", "S100A8",
            "S100A9", "CSF1R", "ITGAM", "CD68", "FCGR3A"), collapse = "\t"),
    paste(c("NK cells", "-", "NKG7", "GNLY", "KLRD1", "KLRF1", "PRF1",
            "GZMB", "NCR1", "KLRC1", "FGFBP2", "SPON2"), collapse = "\t")
  ),
  gmt
)
hc <- hc_functional_enrichment(
  hc,
  gene_sets = character(),
  custom_gmt_files = c(CellTypes = gmt)
)
```

hc_gene_to_cluster	<i>Gene-to-module table (S4 API)</i>
--------------------	--------------------------------------

Description

Returns the mapping of every network gene to the module it was assigned to.

Usage

```
hc_gene_to_cluster(hc)
```

Arguments

hc A HCoCenaExperiment.

Value

A data frame with two columns: gene (gene symbol) and color (module colour). Genes that were not assigned to any module carry the colour "white".

Examples

```
hc <- hc_example_data("clustered")
head(hc_gene_to_cluster(hc))
```

hc_get_meta_time_col_order	<i>Get column order for meta-cluster x timepoint heatmaps</i>
----------------------------	---

Description

Returns the column-order vector stored by hc_add_meta_time_grouping().

Usage

```
hc_get_meta_time_col_order(
  hc,
  slot_name = "longitudinal_endotypes",
  layer = NULL
)
```

Arguments

hc A HCoCenaExperiment.

slot_name Satellite slot containing meta_time_grouping, or a base prefix such as "longitudinal_endotypes" that resolves matching suffixed slots.

layer Layer index, layer id/name, or NULL for combined order across all resolved layers.

Value

Character vector with heatmap column order as expected by `hc_change_grouping_parameter()`.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc
hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
  method = "kmeans",
  consensus = TRUE,
  seed = 1
)
hc <- hc_add_meta_time_grouping(hc, donor_col = "donor", time_col = "timepoint")
hc_get_meta_time_col_order(hc)
```

hc_get_module_scores	<i>get module scores</i>
----------------------	--------------------------

Description

Module scores (S4 API)

Usage

```
hc_get_module_scores(hc, save = TRUE)
```

Arguments

hc	A HCoCenaExperiment.
save	Logical. Write the module-score box plot to PDF. Default is TRUE.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_get_module_scores(hc)
```

hc_graph	<i>Access the integrated network of an HCoCenaExperiment</i>
----------	--

Description

Access the integrated network of an HCoCenaExperiment

Usage

```
hc_graph(x)

## S4 method for signature 'HCoCenaExperiment'
hc_graph(x)
```

Arguments

x An HCoCenaExperiment.

Value

The integrated network as an igraph object, or NULL if hc_build_integrated_network() has not been run yet.

Examples

```
hc <- hc_example_data("clustered")
g <- hc_graph(hc)
igraph::vcount(g)
```

hc_highlight_geneset	<i>Highlight gene set in network (S4 API)</i>
----------------------	---

Description

Re-plots the integrated network with a gene set of interest highlighted.

Usage

```
hc_highlight_geneset(
  hc,
  gene_set,
  name = NULL,
  col = "black",
  label_offset = 3,
  plot = TRUE,
  save = TRUE
)
```

Arguments

hc	A HCoCenaExperiment.
gene_set	Character vector of gene symbols to highlight.
name	Optional name used for the plot title and output file.
col	Colour used for the highlighted genes. Default is "black".
label_offset	Distance between a node and its label. Default is 3.
plot	Logical. Draw the network. Default is TRUE.
save	Logical. Write the network to PDF. Default is TRUE.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_highlight_geneset(hc, gene_set = c("CD3D", "CD19"))
```

hc_import_clusters	<i>Import an external gene-to-cluster assignment</i>
--------------------	--

Description

Replaces the current module assignment with one read from a file, e.g. a clustering exported earlier with [hc_export_clusters\(\)](#).

Usage

```
hc_import_clusters(hc, file, sep = "\t", header = TRUE)
```

Arguments

hc	A HCoCenaExperiment.
file	Path to the gene-to-cluster file. Column 1 holds gene symbols, column 2 the cluster each gene belongs to.
sep	Field separator of the file. Default is "\t".
header	Logical. Whether the file carries a header row. Default TRUE.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
gtc <- hc_gene_to_cluster(hc)
f <- tempfile(fileext = ".tsv")
utils::write.table(gtc, f, sep = "\t", row.names = FALSE, quote = FALSE)
hc <- hc_import_clusters(hc, file = f)
```

hc_import_layout_from_cytoscape

Import layout from Cytoscape (S4 API)

Description

Import layout from Cytoscape (S4 API)

Usage

```
hc_import_layout_from_cytoscape(hc)
```

Arguments

hc	A HCoCenaExperiment.
----	----------------------

Value

Updated HCoCenaExperiment.

Examples

```
# Needs a running Cytoscape instance reachable through RCy3.
cytoscape_up <- requireNamespace("RCy3", quietly = TRUE) &&
  isTRUE(tryCatch({
    RCy3::cytoscapePing()
    TRUE
  }, error = function(e) FALSE))
```

```

if (cytoscape_up) {
  hc <- hc_example_data("clustered")
  hc <- hc_export_to_cytoscape(hc, name = "hcocena example")
  # ... arrange the network in Cytoscape, then:
  hc <- hc_import_layout_from_cytoscape(hc)
}

```

hc_import_layout_from_local_folder

Import Cytoscape layout from local folder (S4 API)

Description

Import Cytoscape layout from local folder (S4 API)

Usage

```
hc_import_layout_from_local_folder(hc, file)
```

Arguments

hc	A HCoCenaExperiment.
file	Path to the network_layout.csv exported from Cytoscape. If omitted, the current output/save folder is used.

Value

Updated HCoCenaExperiment.

Examples

```

hc <- hc_example_data("clustered")
# A layout CSV as written by hc_import_layout_from_cytoscape(): one row per
# node, named by gene, with x and y coordinates.
g <- hc_graph(hc)
xy <- igraph::layout_with_fr(g)
rownames(xy) <- igraph::V(g)$name
layout_file <- file.path(tempdir(), "network_layout.csv")
utils::write.csv(xy, layout_file)
hc <- hc_import_layout_from_local_folder(hc, file = layout_file)

```

hc_init	<i>Initialize an empty HCoCenaExperiment</i>
---------	--

Description

Initialize an empty HCoCenaExperiment

Usage

```
hc_init()
```

Value

A valid, empty HCoCenaExperiment.

Examples

```
hc <- hc_init()
methods::is(hc, "HCoCenaExperiment")
```

hc_init_save_folder	<i>Creates a Save Folder</i>
---------------------	------------------------------

Description

A folder with the given name is created in the output directory. All analysis outputs will be saved to this folder.

Usage

```
hc_init_save_folder(hc, name, use_output_dir = FALSE)
```

Arguments

hc	A HCoCenaExperiment.
name	Folder name. Use "" to write directly into dir_output.
use_output_dir	Boolean. If TRUE, ignore name and use dir_output directly.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("prepared")
hc <- hc_init_save_folder(hc, name = "")
```

hc_integration	<i>Access integration payload from an HCoCenaExperiment</i>
----------------	---

Description

Access integration payload from an HCoCenaExperiment

Usage

```
hc_integration(x)

## S4 method for signature 'HCoCenaExperiment'
hc_integration(x)
```

Arguments

x An HCoCenaExperiment.

Value

A HCoCenaIntegration object.

Examples

```
hc <- hc_init()
class(hc_integration(hc))
```

hc_layer_results	<i>Access layer results from an HCoCenaExperiment</i>
------------------	---

Description

Access layer results from an HCoCenaExperiment

Usage

```
hc_layer_results(x)

## S4 method for signature 'HCoCenaExperiment'
hc_layer_results(x)
```

Arguments

x An HCoCenaExperiment.

Value

A `S4Vectors::SimpleList`.

Examples

```
hc <- hc_init()
class(hc_layer_results(hc))
```

hc_list_celltype_databases

List Enrichr cell-type related databases

Description

Lists Enrichr libraries and optionally filters for cell/tissue related names.

Usage

```
hc_list_celltype_databases(
  pattern = NULL,
  include_all = FALSE,
  ignore_case = TRUE
)
```

Arguments

pattern	Optional regular expression applied to database names.
include_all	Logical. If FALSE (default), applies a built-in cell-type/tissue oriented filter before pattern.
ignore_case	Logical. Passed to <code>base::grepl()</code> .

Value

A data frame with Enrichr library metadata.

Examples

```
hc_list_celltype_databases()
```

hc_list_llm_models	<i>List Available LLM Models</i>
--------------------	----------------------------------

Description

Queries the model-listing endpoint of each selected provider and returns the available models as a data frame. Supports the same providers as `hc_llm_enrichment()`: Gemini, Claude (Anthropic), OpenAI/ChatGPT, and a local OpenAI-compatible server (vLLM).

Usage

```
hc_list_llm_models(
  provider = "all",
  api_key = NULL,
  base_url = NULL,
  timeout_sec = 30
)
```

Arguments

provider	Character vector of providers to query. One or more of "gemini", "claude" ("anthropic"), "openai" ("chatgpt"), and "vllm" ("local"); or "all" (default), which queries the three cloud providers plus the local server when base_url is supplied.
api_key	Optional API key. Only applied when a single provider is requested; otherwise keys are taken from the per-provider environment variables (GEMINI_API_KEY, ANTHROPIC_API_KEY, OPENAI_API_KEY, VLLM_API_KEY).
base_url	Base URL of the local OpenAI-compatible server. Required for "vllm"; falls back to VLLM_BASE_URL or "http://localhost:8000/v1".
timeout_sec	Per-request timeout in seconds. Default 30.

Value

A data frame with columns provider, model, and info. Providers that cannot be reached (e.g. a missing API key) are skipped with a warning rather than aborting the whole call.

Examples

```
# Contacts the provider APIs; each provider needs its API key set in the
# environment (or a reachable server for "vllm").

if (nzchar(Sys.getenv("GEMINI_API_KEY"))) {
  hc_list_llm_models("gemini")
}
if (nzchar(Sys.getenv("OPENAI_API_KEY"))) {
  hc_list_llm_models("openai")
}
```

 hc_llm_enrichment

LLM-based interpretation of modules or feature sets

Description

Asks a large language model what biological program each module (or any other set of genes, proteins or other features) most likely represents. This is an AI-assisted interpretation, not a statistical test.

Usage

```
hc_llm_enrichment(
  hc = NULL,
  genes = NULL,
  modules = "all",
  system_prompt = NULL,
  context = NULL,
  provider = c("claude", "openai", "gemini", "local", "manual"),
  api_key = NULL,
  model = NULL,
  base_url = NULL,
  evidence = "enrichment",
  evidence_text = NULL,
  evidence_top = 10,
  evidence_qval = 0.05,
  rag_url = NULL,
  rag_top_k = 10,
  slot_name = "llm_enrichment",
  verbose = TRUE
)
```

Arguments

hc	Optional HCoCenaExperiment. Required to annotate modules, to use stored enrichment results and to store the results; not needed for genes.
genes	Optional feature set(s) to interpret instead of modules: a character vector (one set) or a named list of character vectors.
modules	Modules of hc to interpret, or "all" (default). Needs the module table created by hc_plot_cluster_heatmap() . Ignored when genes is given.
system_prompt	Optional instruction that tells the model who it is and how to answer, e.g. "You are an expert proteomics analyst.". A neutral default suited to any kind of features is used when NULL.
context	Optional free text describing the study, e.g. tissue, disease, platform or time point. Used for all fields except general_processes.

provider	"claude", "openai", "gemini", "local" (vLLM or any OpenAI-compatible server) or "manual" (no API key; copy and paste via the web chat of any provider).
api_key	API key. If NULL, it is read from ANTHROPIC_API_KEY, OPENAI_API_KEY, GEMINI_API_KEY or VLLM_API_KEY. Not used for "manual".
model	Optional model name. Defaults to "claude-sonnet-4-6", "gpt-4o-mini", "gemini-2.5-pro" or "Qwen/Qwen2.5-VL-32B-Instruct" (local). For "manual" it is only recorded with the result.
base_url	Server address for provider = "local". If NULL, read from VLLM_BASE_URL, falling back to "http://localhost:8000/v1".
evidence	Evidence for grounded_processes: "enrichment" (default; needs hc_functional_enrichment()), any combination with "upstream" (needs hc_upstream_inference()) and "rag" (literature passages, needs rag_url), "all", or "none" to skip this request.
evidence_text	Optional evidence supplied directly, as a named list or named character vector (names = set names). Added to the evidence above; the only way to give evidence together with genes.
evidence_top	Maximum number of enriched terms and of regulators per set. Default 10.
evidence_qval	Adjusted p-value cutoff for enriched terms. Default 0.05.
rag_url	Retrieval endpoint of a DoRAG server for "rag". If NULL, read from the environment variable HCOCEANA_RAG_URL.
rag_top_k	Number of literature passages per set. Default 10.
slot_name	Name under which results are stored in hc. Default "llm_enrichment".
verbose	Logical; print progress messages. Default TRUE.

Details

Every set is interpreted in up to three separate requests, so that each answer sees only the input it is meant to see:

- general_processes: from the features alone, without context;
- contextual_state and key_regulators: from the features and the context;
- grounded_processes and supporting_evidence: from the features, the context and the evidence - by default the significant terms of [hc_functional_enrichment\(\)](#); optionally also the regulators of [hc_upstream_inference\(\)](#) and literature passages from a DoRAG server.

[hc_plot_llm_enrichment\(\)](#) draws one plot per field, so the plots can be compared directly: what the model makes of the genes alone, with context, and with statistical evidence. Without stored enrichment results the third request is left out (with a message).

The requests can be sent in two ways:

- through an API (provider = "claude", "openai", "gemini", or "local" for an OpenAI-compatible vLLM server), which needs an API key for the hosted providers;
- without an API key (provider = "manual"): one prompt per request type is prepared (the first is copied to the clipboard); paste each into a new chat of any provider and read every answer back with [hc_llm_import\(\)](#). The results are stored exactly like API results.

Results are stored in `hc_satellite(hc, slot_name)` (one entry per set) and `hc_satellite(hc, paste0(slot_name, "_summary"))` (a table).

Value

With `hc`: the updated `HCoCenaExperiment` (for "manual", with the pending request stored). Without `hc`: a data frame with one row per set (for "manual", an `hc_llm_request` object to pass to `hc_llm_import()`).

See Also

`hc_llm_import()`, `hc_plot_llm_enrichment()`, `hc_list_llm_models()`

Examples

```
# Without an API key: build the requests, answer them in a web chat, import.
req <- hc_llm_enrichment(
  genes = list(IFN = c("STAT1", "IRF7", "CXCL10", "GBP1", "IFI44L")),
  context = "Whole blood of patients with acute viral infection",
  evidence_text = list(IFN = "Enriched: type I interferon signaling (q = 1e-6)"),
  provider = "manual",
  verbose = FALSE
)
length(req$parts) # genes only / with context / with evidence
cat(substr(req$parts[[1]], 1, 300))

# The answers copied from the web chat (here written by hand):
answer <- paste0(
  '{"request_id": "", req$id, "", "results": [{"set": "IFN", ',
  '"general_processes": "type I interferon signaling / antiviral response",',
  '"contextual_state": "interferon-high antiviral state",',
  '"key_regulators": "STAT1 / IRF7 / IRF9",',
  '"grounded_processes": "type I interferon response",',
  '"supporting_evidence": "type I interferon signaling"}]}'
)
hc_llm_import(req, text = answer, verbose = FALSE)

# Through an API (needs a key):
if (nzchar(Sys.getenv("ANTHROPIC_API_KEY"))) {
  hc_llm_enrichment(
    genes = c("STAT1", "IRF7", "CXCL10", "GBP1", "IFI44L"),
    provider = "claude"
  )
}
```

hc_llm_import	<i>Import an answer from a provider's web chat</i>
---------------	--

Description

Reads an answer to a request created with `hc_llm_enrichment(provider = "manual")`, checks that it belongs to that request, and stores it exactly like an API result. Markdown code fences, text around the JSON and typographic quotes are tolerated.

Usage

```
hc_llm_import(
  x,
  text = NULL,
  file = NULL,
  model = NULL,
  slot_name = "llm_enrichment",
  verbose = TRUE
)
```

Arguments

<code>x</code>	The HCoCenaExperiment returned by <code>hc_llm_enrichment(provider = "manual")</code> , or the <code>hc_llm_request</code> object it returned when no <code>hc</code> was given.
<code>text</code>	Optional answer text. If <code>NULL</code> and <code>file</code> is <code>NULL</code> , the clipboard is read (needs the <code>clipr</code> package and a clipboard, i.e. not inside a remote RStudio Server or Docker session).
<code>file</code>	Optional path to a text file containing the answer.
<code>model</code>	Optional name of the model used in the web chat, recorded with the result.
<code>slot_name</code>	Slot used in <code>hc_llm_enrichment()</code> . Default <code>"llm_enrichment"</code> .
<code>verbose</code>	Logical; print progress messages. Default <code>TRUE</code> .

Details

A request has one part per request type (features only, with context, with evidence), and long requests are split further. Import every answer with this function; after each import the next open part is copied to the clipboard. An answer may also contain the fields of several parts.

Value

For an HCoCenaExperiment, the updated object. For an `hc_llm_request`, a data frame with one row per set answered so far.

See Also

[hc_llm_enrichment\(\)](#)

Examples

```
req <- hc_llm_enrichment(
  genes = list(IFN = c("STAT1", "IRF7", "CXCL10")),
  provider = "manual", verbose = FALSE
)
answer <- paste0(
  "Here is the result:\n```\njson\n",
  '{"request_id": "", req$id: "", "results": [{"set": "IFN", ',
  '"general_processes": "interferon signaling", ',
  '"contextual_state": "antiviral state", "key_regulators": "STAT1"}]}' ,
  "\n```\n"
)
hc_llm_import(req, text = answer, model = "web chat", verbose = FALSE)
```

```
hc_longitudinal_endotype_clustering
```

```
Cluster donors into longitudinal endotypes
```

Description

Clusters donors based on module-time features produced by [hc_longitudinal_module_means\(\)](#). Candidate cluster numbers are scored via the Calinski-Harabasz index.

Usage

```
hc_longitudinal_endotype_clustering(
  hc,
  input_slot = "longitudinal_module_means",
  output_slot = "longitudinal_endotypes",
  k = 2:6,
  method = c("kmeans", "hclust"),
  scale_features = TRUE,
  nstart = 50,
  seed = 42,
  na_impute = c("median", "zero"),
  endotype_prefix = "E",
  module_k = NULL,
  module_method = c("kmeans", "hclust"),
  module_scale_features = TRUE,
  module_nstart = 30,
  module_seed = 42,
  module_na_impute = c("median", "zero")
)
```

Arguments

hc A HCoCenaExperiment.

input_slot	Satellite slot produced by hc_longitudinal_module_means().
output_slot	Satellite slot to store clustering results.
k	Integer vector of candidate cluster counts.
method	One of "kmeans" or "hclust".
scale_features	Logical. If TRUE, z-score features before clustering.
nstart	Number of random starts for kmeans.
seed	Random seed.
na_impute	One of "median" or "zero" for feature NA imputation.
endotype_prefix	Prefix for final endotype labels.
module_k	Candidate cluster counts for module-wise trajectory clustering.
module_method	One of "kmeans" or "hclust" for module-wise clustering.
module_scale_features	Logical. If TRUE, z-score module trajectory features before module-wise clustering.
module_nstart	Number of random starts for module-wise kmeans.
module_seed	Random seed for module-wise clustering.
module_na_impute	One of "median" or "zero" for module-wise NA imputation.

Value

Updated HCoCenaExperiment with endotype clustering in hc@satellite[[output_slot]].

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_longitudinal_module_means(hc, donor_col = "donor", time_col = "timepoint")
hc <- hc_longitudinal_endotype_clustering(hc, k = 2)
```

hc_longitudinal_meta_clustering

Meta-cluster donors based on longitudinal CAP features

Description

Clusters donors on CAP features (or alternative longitudinal feature matrices), providing CH-based model selection and PCA/UMAP embeddings.

Usage

```

hc_longitudinal_meta_clustering(
  hc,
  slot_name = "longitudinal_endotypes",
  output_slot = slot_name,
  feature_source = c("cap_matrix", "module_cluster_matrix", "feature_matrix_used"),
  k = 2:8,
  method = c("kmeans", "hclust"),
  scale_features = TRUE,
  nstart = 50,
  seed = 42,
  na_impute = c("median", "zero"),
  cluster_prefix = "MC",
  consensus = TRUE,
  consensus_runs = 200,
  consensus_sample_fraction = 0.8,
  consensus_feature_fraction = 0.8,
  consensus_linkage = "average",
  compute_umap = TRUE,
  umap_neighbors = 15,
  umap_min_dist = 0.3
)

```

Arguments

<code>hc</code>	A HCoCenaExperiment.
<code>slot_name</code>	Satellite slot containing longitudinal outputs.
<code>output_slot</code>	Satellite slot to write results to (default in-place).
<code>feature_source</code>	One of "cap_matrix", "module_cluster_matrix", or "feature_matrix_used".
<code>k</code>	Candidate meta-cluster numbers.
<code>method</code>	One of "kmeans" or "hclust".
<code>scale_features</code>	Logical. If TRUE, z-score features before clustering.
<code>nstart</code>	Number of random starts for kmeans.
<code>seed</code>	Random seed.
<code>na_impute</code>	One of "median" or "zero".
<code>cluster_prefix</code>	Prefix for meta-cluster labels.
<code>consensus</code>	Logical. If TRUE, run consensus clustering with repeated subsampling for each candidate k and select the best k via consensus separation.
<code>consensus_runs</code>	Number of consensus resampling runs per candidate k.
<code>consensus_sample_fraction</code>	Fraction of donors sampled per consensus run.
<code>consensus_feature_fraction</code>	Fraction of features sampled per consensus run.

consensus_linkage Linkage method for final hierarchical clustering on the consensus distance (1 - consensus).

compute_umap Logical. If TRUE, compute UMAP coordinates when uwot or umap is available.

umap_neighbors UMAP neighborhood size.

umap_min_dist UMAP minimum distance.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc
hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
  method = "kmeans",
  seed = 1
)
```

hc_longitudinal_module_cap

Compute module-level CAP matrix from repeated trajectory clustering

Description

CAP (cluster assignment proportions) quantifies for each donor the proportion of repeated clustering runs assigned to each module-specific cluster. This is the generalized, data-agnostic analogue of the CAP concept used in the legacy longitudinal scripts.

Usage

```
hc_longitudinal_module_cap(
  hc,
  slot_name = "longitudinal_endotypes",
  output_slot = slot_name,
  runs = 50,
  method = c("kmeans", "hclust"),
  nstart = 30,
  seed = 42,
  scale_features = TRUE,
  na_impute = c("median", "zero"),
  jitter_sd = 0
)
```

Arguments

hc	A HCoCenaExperiment.
slot_name	Satellite slot containing outputs from hc_longitudinal_endotype_clustering().
output_slot	Satellite slot to write results to. Defaults to slot_name (in-place update).
runs	Number of repeated clustering runs per module (for kmeans).
method	One of "kmeans" or "hclust" for repeated module clustering.
nstart	kmeans random starts per run.
seed	Random seed.
scale_features	Logical. If TRUE, z-score module trajectories before repeated clustering.
na_impute	One of "median" or "zero" for NA imputation in module trajectory matrices.
jitter_sd	Optional Gaussian jitter SD added to features per run (kmeans only) to probe local assignment stability.

Value

Updated HCoCenaExperiment with CAP outputs in hc@satellite[[output_slot]].

Examples

```
hc <- hc_example_data("clustered")
long <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)
hc <- long$hc
hc <- hc_longitudinal_module_cap(hc, runs = 1)
```

 hc_longitudinal_module_means

Build longitudinal module means per donor and timepoint

Description

Computes module-level mean expression trajectories from the clustered modules in an HCoCenaExperiment. This creates a donor-by-(module x timepoint) feature matrix that can be used for longitudinal endo-type discovery.

Usage

```
hc_longitudinal_module_means(
  hc,
  donor_col = "donor",
  time_col = "time_point",
  layer = NULL,
  group_col = NULL,
  modules = NULL,
  use_module_labels = TRUE,
  time_levels = NULL,
  aggregate_fun = function(x) base::mean(x, na.rm = TRUE),
  value_label = NULL,
  impute_missing = c("linear", "locf", "none"),
  min_genes_per_module = 5,
  slot_name = "longitudinal_module_means"
)
```

Arguments

hc	A HCoCenaExperiment.
donor_col	Annotation column holding donor/patient IDs.
time_col	Annotation column holding ordered timepoint labels.
layer	Layer to use (index, layer id, or layer name). Defaults to the first available layer.
group_col	Optional annotation column (e.g. outcome group) to carry.
modules	Optional subset of modules (module labels or color names).
use_module_labels	Logical. If TRUE, use M-style labels from hc_plot_cluster_heatmap() when available; otherwise use color names.
time_levels	Optional explicit timepoint order.
aggregate_fun	Function used to aggregate technical replicates per donor/timepoint/module (default mean).
value_label	Optional y-axis label base (e.g. "Module mean GFC"). If NULL, an automatic label is inferred from the selected layer id.

impute_missing One of "none", "linear", "locf" to impute missing donor-timepoint values within each donor/module trajectory.

min_genes_per_module Minimum number of overlapping genes required to keep a module.

slot_name Name of the satellite slot to store outputs.

Value

Updated HCoCenaExperiment with longitudinal data in hc@satellite[[slot_name]].

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_longitudinal_module_means(hc, donor_col = "donor", time_col = "timepoint")
```

hc_longitudinal_step1_module_donor

Longitudinal step 1: module/donor clustering

Description

Computes longitudinal module means, then clusters donors separately within each module directly on the donor-by-time trajectory matrix. The downstream quick steps remain unchanged and can be run via [hc_longitudinal_step2_meta_clustering\(\)](#), [hc_longitudinal_step3_meta_module_trajectories\(\)](#), and [hc_plot_longitudinal_enrichment_meta_waves\(\)](#).

Usage

```
hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "Subject",
  time_col = "Time_token",
  layer = NULL,
  time_levels,
  k = 2:6,
  method = c("kmeans", "hclust"),
  nstart = 50,
  cap_runs = 50,
  impute = TRUE,
  impute_method = "rfcont",
  ntree = 10,
  min_cluster_fraction = 0.1,
  score_method = c("calinski_harabasz", "tot_withinss"),
  scale_features = FALSE,
  na_impute = c("median", "zero"),
  cap_scale_features = scale_features,
  cap_na_impute = na_impute,
```

```

    cap_jitter_sd = 0,
    seed = 42,
    means_slot = "longitudinal_module_means",
    output_slot = "longitudinal_endotypes"
  )

```

Arguments

hc	A HCoCenaExperiment.
donor_col	Annotation column containing donor IDs.
time_col	Annotation column containing timepoint labels.
layer	Optional layer index, layer id, layer name, or "all". NULL uses the first available layer.
time_levels	Required explicit timepoint order vector.
k	Candidate donor-trajectory cluster numbers per module.
method	One of "kmeans" or "hclust" for direct clustering on the donor-by-time matrix.
nstart	Number of random starts for direct kmeans.
cap_runs	Number of repeated clustering runs used to build CAP.
impute	Logical. If TRUE, impute missing donor-time values before direct clustering.
impute_method	Missing-value method passed to mice.
ntree	Number of trees for rfcont imputation.
min_cluster_fraction	Minimum allowed fraction of donors in the smallest module cluster.
score_method	One of "calinski_harabasz" or "tot_withinss" for direct selection.
scale_features	Logical. If TRUE, z-score timepoint columns before clustering.
na_impute	One of "median" or "zero" for final NA handling after optional imputation.
cap_scale_features	Logical. If TRUE, z-score features for CAP reruns.
cap_na_impute	One of "median" or "zero" for CAP feature handling.
cap_jitter_sd	Optional Gaussian jitter SD added per CAP rerun when method = "kmeans".
seed	Random seed.
means_slot	Satellite slot for module means.
output_slot	Satellite slot for endotype outputs.

Value

A list with updated hc, step plots, and diagnostics. When layer = "all" (or multiple layers are supplied), returns per-layer plot and diagnostic lists keyed by layer id.

Examples

```

hc <- hc_example_data("clustered")
res <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)
hc <- res$hc

```

```
hc_longitudinal_step1_module_donor_direct
```

Longitudinal step 1: module/donor clustering with direct trajectories

Description

Computes longitudinal module means, then clusters donors separately within each module directly on the donor-by-time trajectory matrix, especially for short trajectories with only a few timepoints.

Usage

```

hc_longitudinal_step1_module_donor_direct(
  hc,
  donor_col = "Subject",
  time_col = "Time_token",
  layer = NULL,
  time_levels,
  k = 2:6,
  method = c("kmeans", "hclust"),
  nstart = 50,
  cap_runs = 50,
  impute = TRUE,
  impute_method = "rfcont",
  ntree = 10,
  min_cluster_fraction = 0.1,
  score_method = c("calinski_harabasz", "tot_withinss"),
  scale_features = TRUE,
  na_impute = c("median", "zero"),
  cap_scale_features = scale_features,
  cap_na_impute = na_impute,
  cap_jitter_sd = 0,
  seed = 42,

```

```

    means_slot = "longitudinal_module_means_direct",
    output_slot = "longitudinal_endotypes_direct"
  )

```

Arguments

hc	A HCoCenaExperiment.
donor_col	Annotation column containing donor IDs.
time_col	Annotation column containing timepoint labels.
layer	Optional layer index, layer id, layer name, or "all". NULL uses the first available layer.
time_levels	Required explicit timepoint order vector.
k	Candidate donor-trajectory cluster numbers per module.
method	One of "kmeans" or "hclust" for direct clustering on the donor-by-time matrix.
nstart	Number of random starts for direct kmeans.
cap_runs	Number of repeated clustering runs used to build CAP.
impute	Logical. If TRUE, impute missing donor-time values before direct clustering.
impute_method	Missing-value method passed to mice. Use "rfcont" to match the previous workflow most closely. For "rfcont", CALIBERrfimpute must be installed; it is loaded automatically.
ntree	Number of trees for rfcont imputation.
min_cluster_fraction	Minimum allowed fraction of donors in the smallest module cluster.
score_method	One of "calinski_harabasz" or "tot_withinss" for selecting the final per-module k.
scale_features	Logical. If TRUE, z-score timepoint columns before clustering.
na_impute	One of "median" or "zero" for final NA handling after optional imputation.
cap_scale_features	Logical. If TRUE, z-score features for the CAP reruns.
cap_na_impute	One of "median" or "zero" for CAP feature handling.
cap_jitter_sd	Optional Gaussian jitter SD added per CAP rerun when method = "kmeans".
seed	Random seed.
means_slot	Satellite slot for module means.
output_slot	Satellite slot for direct endotype outputs.

Value

A list with updated hc, step plots, and diagnostics. When layer = "all" (or multiple layers are supplied), returns per-layer plot and diagnostic lists keyed by layer id.

Examples

```

hc <- hc_example_data("clustered")
res <- hc_longitudinal_step1_module_donor_direct(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)
hc <- res$hc

```

```
hc_longitudinal_step2_meta_clustering
```

Longitudinal step 2: meta-clustering

Description

Runs the exact CAP -> PCA -> graph -> Leiden donor meta-clustering and returns the PCA/UMAP embeddings.

Usage

```

hc_longitudinal_step2_meta_clustering(
  hc,
  slot_name = "longitudinal_endotypes",
  dimensions = 4,
  graph_method = c("knn", "snn"),
  knn_method = c("annoy", "rann"),
  graph_k = 7,
  resolution = 0.4,
  leiden_method = "RBConfigurationVertexPartition",
  cluster_prefix = "MC",
  compute_umap = TRUE,
  umap_neighbors = 15,
  umap_min_dist = 0.3,
  seed = 42
)

```

Arguments

hc	A HCoCenaExperiment.
slot_name	Satellite slot containing step 1 outputs. Use a concrete slot name, a shared prefix such as "longitudinal_endotypes" to process matching suffixed slots, or "all" to run over every compatible step 1 slot in hc@satellite.

dimensions	Number of PCA dimensions used to build the donor graph.
graph_method	Graph type used before Leiden clustering.
knn_method	Nearest-neighbor backend used for graph construction.
graph_k	Number of neighbors used for the donor graph.
resolution	Leiden resolution parameter.
leiden_method	Leiden partition method.
cluster_prefix	Prefix for meta-cluster labels.
compute_umap	Logical. If TRUE, compute an additional UMAP from the PCA subspace used for graph clustering.
umap_neighbors	UMAP neighborhood size.
umap_min_dist	UMAP minimum distance.
seed	Random seed.

Value

A list with updated hc, step plots, and diagnostics. When multiple slots are processed, plots and diagnostics are returned as named lists keyed by layer/slot, plus a slot_info table describing the resolved slots.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc
res <- hc_longitudinal_step2_meta_clustering(hc, graph_k = 5, dimensions = 2)
hc <- res$hc
```

hc_longitudinal_step3_meta_module_trajectories

Longitudinal step 3: module trajectories by meta-cluster

Description

Plots module trajectories after meta-clustering using donor and meta-cluster mean waves.

Usage

```
hc_longitudinal_step3_meta_module_trajectories(
  hc,
  slot_name = "longitudinal_endotypes",
  facet_ncol = 4,
  free_y = TRUE,
  square_panels = TRUE,
  value_mode = c("scaled_mean_vst", "expression"),
  value_range = c(-2, 2)
)
```

Arguments

hc	A HCoCenaExperiment.
slot_name	Satellite slot containing meta-clustering outputs. Use a concrete slot name, a shared prefix such as "longitudinal_endotypes" to process matching suffixed slots, or "all" to run over every compatible step 2 slot in hc@satellite.
facet_ncol	Number of facet columns.
free_y	Logical. If TRUE, use free y-scales per module.
square_panels	Logical. If TRUE, draw square panels.
value_mode	One of "scaled_mean_vst" or "expression".
value_range	Optional numeric vector of length 2 for the y-axis range when value_mode = "scaled_mean_vst". Default is c(-2, 2). Use NULL for automatic scaling.

Value

A list with unchanged hc and the step 3 plot. When multiple slots are processed, plots is returned as a named list keyed by layer/slot, plus a slot_info table describing the resolved slots.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc
hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
```

```

    method = "kmeans",
    consensus = TRUE,
    seed = 1
  )
  res <- hc_longitudinal_step3_meta_module_trajectories(hc)

```

hc_longitudinal_workflow_direct

Convenience longitudinal workflow using the direct step-1 backend

Description

Runs the direct donor-by-time step 1 backend and then reuses the existing step 2 and step 3 quick wrappers on the resulting output slot family.

Usage

```

hc_longitudinal_workflow_direct(
  hc,
  donor_col = "Subject",
  time_col = "Time_token",
  layer = NULL,
  time_levels,
  k = 2:6,
  method = c("kmeans", "hclust"),
  nstart = 50,
  cap_runs = 50,
  impute = TRUE,
  impute_method = "rfcont",
  ntree = 10,
  min_cluster_fraction = 0.1,
  score_method = c("calinski_harabasz", "tot_withinss"),
  scale_features = TRUE,
  na_impute = c("median", "zero"),
  cap_scale_features = scale_features,
  cap_na_impute = na_impute,
  cap_jitter_sd = 0,
  seed = 42,
  means_slot = "longitudinal_module_means_direct",
  output_slot = "longitudinal_endotypes_direct"
)

```

Arguments

hc	A HCoCenaExperiment.
donor_col	Annotation column containing donor IDs.
time_col	Annotation column containing timepoint labels.

layer	Optional layer index, layer id, layer name, or "all". NULL uses the first available layer.
time_levels	Required explicit timepoint order vector.
k	Candidate donor-trajectory cluster numbers per module.
method	One of "kmeans" or "hclust" for direct clustering on the donor-by-time matrix.
nstart	Number of random starts for direct kmeans.
cap_runs	Number of repeated clustering runs used to build CAP.
impute	Logical. If TRUE, impute missing donor-time values before direct clustering.
impute_method	Missing-value method passed to mice. Use "rfcont" to match the previous workflow most closely. For "rfcont", CALIBERrfimpute must be installed; it is loaded automatically.
ntree	Number of trees for rfcont imputation.
min_cluster_fraction	Minimum allowed fraction of donors in the smallest module cluster.
score_method	One of "calinski_harabasz" or "tot_withinss" for selecting the final per-module k.
scale_features	Logical. If TRUE, z-score timepoint columns before clustering.
na_impute	One of "median" or "zero" for final NA handling after optional imputation.
cap_scale_features	Logical. If TRUE, z-score features for the CAP reruns.
cap_na_impute	One of "median" or "zero" for CAP feature handling.
cap_jitter_sd	Optional Gaussian jitter SD added per CAP rerun when method = "kmeans".
seed	Random seed.
means_slot	Satellite slot for module means.
output_slot	Satellite slot for direct endotype outputs.

Value

A list with updated hc, nested plots, and per-step diagnostics.

Examples

```
hc <- hc_example_data("clustered")
res <- hc_longitudinal_workflow_direct(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)
hc <- res$hc
```

hc_mae	<i>Access MultiAssayExperiment from an HCoCenaExperiment</i>
--------	--

Description

Access MultiAssayExperiment from an HCoCenaExperiment

Usage

```
hc_mae(x)

## S4 method for signature 'HCoCenaExperiment'
hc_mae(x)
```

Arguments

x An HCoCenaExperiment.

Value

A MultiAssayExperiment::MultiAssayExperiment object.

Examples

```
hc <- hc_init()
class(hc_mae(hc))
```

hc_merge_clusters	<i>Merge similar modules</i>
-------------------	------------------------------

Description

Cuts the dendrogram of the module heatmap to merge modules with similar expression patterns. Run with save = FALSE first to preview a given k; save = TRUE overwrites the current module assignment and cannot be undone.

Usage

```
hc_merge_clusters(
  hc,
  k = "auto",
  save = TRUE,
  method = "complete",
  k_min = 2,
  k_max = NULL,
  auto_parsimony_penalty = 1e-04,
  verbose = TRUE
)
```

Arguments

hc	A HCoCenaExperiment.
k	Either an integer (the number of merged modules) or "auto" (default) to pick a data-driven value from the average silhouette score across candidate cuts.
save	Logical. If FALSE, only preview the merge. If TRUE, replace the current module assignment. Default is TRUE.
method	Agglomeration method for the dendrogram. Default "complete".
k_min	Smallest candidate k considered when k = "auto". Default 2.
k_max	Largest candidate k considered when k = "auto". Defaults to min(10, number of modules - 1).
auto_parsimony_penalty	Small penalty applied in auto mode to prefer fewer modules when silhouette scores are nearly tied. Default 1e-04.
verbose	Logical. Print the selected k and selection diagnostics.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_plot_cluster_heatmap(hc, file_name = FALSE)
hc <- hc_merge_clusters(hc, k = 2, save = FALSE)
```

hc_meta_correlation_cat

Correlate categorical metadata with modules (S4 API)

Description

Correlate categorical metadata with modules (S4 API)

Usage

```
hc_meta_correlation_cat(hc, meta, set, p_val = 0.05, padj = "BH")
```

Arguments

hc	A HCoCenaExperiment.
meta	A single string. Name of the categorical annotation column to correlate with the module expression patterns.
set	An integer. Number of the dataset/layer to use.
p_val	Maximum adjusted p-value for a correlation to count as significant. Default is 0.05. Non-significant cells are shown in grey.
padj	Multiple-testing correction method passed to <code>stats::p.adjust()</code> . Default is "BH".

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc_meta_correlation_cat(hc, set = 1, meta = "batch")
```

hc_meta_correlation_num

Correlate numeric metadata with modules (S4 API)

Description

Correlates the mean expression of each module's genes per sample with one or more numeric annotation columns, and shows the result as a heatmap. Non-significant correlations are left blank.

Usage

```
hc_meta_correlation_num(hc, set, meta, p_val = 0.05, padj = "BH")
```

Arguments

hc	A HCoCenaExperiment.
set	An integer. Number of the dataset/layer to use.
meta	Character vector of numeric annotation column names to correlate with the module expression patterns.
p_val	Maximum adjusted p-value for a correlation to count as significant. Default is 0.05.
padj	Multiple-testing correction method passed to <code>stats::p.adjust()</code> . Default is "BH".

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc_meta_correlation_num(hc, set = 1, meta = "age")
```

hc_meta_plot	<i>meta plot</i>
--------------	------------------

Description

Meta-data plotting (S4 API)

Usage

```
hc_meta_plot(
  hc,
  set,
  group_col = NULL,
  meta_col = NULL,
  type = "cat",
  cols = NULL
)
```

Arguments

hc	A HCoCenaExperiment.
set	An integer. Number of the dataset/layer to inspect.
group_col	Annotation column used to group the samples, e.g. the timepoint or condition.
meta_col	Annotation column whose distribution is plotted.
type	Either "cat" (default) for categorical meta_col or "num" for continuous ones.
cols	Optional named vector of colours for the groups.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("prepared")
hc <- hc_meta_plot(hc, set = 1, group_col = "group", meta_col = "batch")
```

 hc_module_condition_significance

Test module differences between conditions (S4 API)

Description

Test module differences between conditions (S4 API)

Usage

```
hc_module_condition_significance(
  hc,
  set = "all",
  condition_col = NULL,
  donor_col = NULL,
  time_col = NULL,
  run_wilcox = TRUE,
  run_limma = TRUE,
  run_lmm = FALSE,
  lmm_include_time = TRUE,
  limma_reference = NULL,
  limma_trend = TRUE,
  standardize_modules = FALSE,
  padj = "BH",
  export_excel = TRUE,
  excel_file = "Module_condition_significance.xlsx",
  slot_name = "module_condition_significance"
)
```

Arguments

hc	A HCoCenaExperiment.
set	Integer vector of layer indices or "all" (default).
condition_col	Column name in annotation files defining conditions. If NULL, uses hcoobject[["global_settings"]][["condition_col"]].
donor_col	Optional donor identifier column in annotation files. Required if run_lmm = TRUE.
time_col	Optional time column in annotation files (used by LMM when available and lmm_include_time = TRUE).
run_wilcox	Logical; run Wilcoxon/Kruskal module-wise test.
run_limma	Logical; run limma pairwise contrasts.
run_lmm	Logical; run linear mixed model (nlme::lme) per module.
lmm_include_time	Logical; include time_col as fixed effect in LMM when available.

limma_reference	Optional reference condition for one-vs-reference limma contrasts. If NULL, all pairwise contrasts are tested.
limma_trend	Logical; passed to <code>limma::eBayes(trend = ...)</code> .
standardize_modules	Logical; z-score each module across the samples of a layer before testing. Raw module means carry the module's absolute expression level, so limma's variance moderation - which borrows information across modules - is dominated by the highly expressed ones, and the reported effect sizes are not comparable between modules. The standardization is applied per layer, so with several layers pooled (<code>set = "all"</code>) it also removes the between-layer offset. With a single layer the rank-based Wilcoxon/Kruskal p-values are unchanged, because z-scoring is monotone within a row; pooled across layers they do change, because the layers are brought onto a common scale first. That is usually what you want when the layers are different tissues or assays, since neither the limma design (<code>~ 0 + condition</code>) nor the LMM (<code>value ~ condition, random ~ 1 donor</code>) carries a layer term. Default FALSE keeps the previous behaviour.
padj	Multiple-testing correction method (passed to <code>p.adjust/limma</code>).
export_excel	Logical; write result tables to Excel in save folder.
excel_file	File name for the Excel export.
slot_name	Satellite slot name for storing results.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_module_condition_significance(
  hc,
  condition_col = "group",
  run_limma = FALSE,
  export_excel = FALSE
)
```

hc_module_gfc_tables *Module GFC per group and per sample*

Description

Returns the Group-Fold-Change values behind the module heatmap as tables: once aggregated per group (the numbers shown in the heatmap) and once resolved per individual sample.

Usage

```

hc_module_gfc_tables(
  hc,
  layer = 1,
  group_col = NULL,
  modules = "all",
  annotate = TRUE,
  export_xlsx = TRUE,
  out_dir = NULL,
  file_name = "Module_GFC_group_and_sample.xlsx",
  verbose = TRUE
)

```

Arguments

hc	A HCoCenaExperiment.
layer	Layer to use, either its index or its name. Defaults to the first layer.
group_col	Annotation column used for grouping. Defaults to the variable_of_interest from the global settings.
modules	Either "all" (default) or a vector of module labels.
annotate	Logical; if TRUE (default), the layer annotation is appended to the long-format sample table so it can be grouped directly.
export_xlsx	Logical; if TRUE (default), all tables are written to an Excel workbook.
out_dir	Target folder for the workbook. If NULL (default), the save folder of hc is used.
file_name	File name of the workbook.
verbose	Logical; print the self-check and the export path.

Details

hCoCena itself only stores the group level (module_gfc_means, modules by groups). This function computes the per-sample counterpart using hCoCena's own GFC definition, so both tables live on the same scale:

1. when data_in_log is TRUE, values are anti-logged (base 2) and all averaging happens on the linear scale,
2. group means (or the raw sample values) are formed per gene,
3. the reference follows the control_keyword global setting, exactly as [hc_run_expression_analysis_2\(\)](#) does. With "none" it is the unweighted mean of the **group** means – not the mean across all samples, which differs whenever the groups are unequally sized. With a control keyword it is the mean of the matching control group, and that group is dropped from the group table just like in the heatmap,
4. foldchange(x, ref) is x / ref when $x \geq \text{ref}$ and $-\text{ref} / x$ otherwise, a signed ratio rather than a difference of logs,
5. values are clamped to $\pm \text{range_GFC}$,
6. the module value is the mean across the genes of that module.

Because step 4 is non-linear and step 5 truncates, **the group table is not the average of the sample table** ($\text{fc}(\text{mean}(x)) \neq \text{mean}(\text{fc}(x))$). Both are correct but answer different questions: `by_group` is the value plotted in the heatmap, `by_sample` shows how uniformly the samples carry it. Use `by_sample` for sample-level statistics and `by_group` for figure labels.

As a self-check the function compares its own `by_group` against the `grp_means` stored by `hCoCena` and reports the largest absolute difference, which is expected to be on the order of $1e-3$ – the rounding `hCoCena` applies per gene.

Value

Updated `HCoCenaExperiment`. The tables are stored in `hc@satellite$module_gfc_tables` as `by_group`, `by_group_long`, `by_sample`, `by_sample_long`, plus reference and settings.

See Also

[hc_plot_cluster_heatmap\(\)](#), which produces the group-level `module_gfc_means` table.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_module_gfc_tables(hc, export_xlsx = FALSE)
hc_satellite(hc, "module_gfc_tables")$by_group
```

hc_network_comparison_1

Compare two gene-to-cluster assignments

Description

Compares two externally stored gene-to-cluster tables and visualises how the modules of the first assignment map onto those of the second.

Usage

```
hc_network_comparison_1(
  hc,
  gtc1_path,
  gtc2_path,
  sep = "\t",
  header = TRUE,
  cellsize = 18
)
```

Arguments

hc	A HCoCenaExperiment.
gtc1_path	Path to the first gene-to-cluster file.
gtc2_path	Path to the second gene-to-cluster file.
sep	Field separator of both files. Default is "\t".
header	Logical. Whether the files carry a header row. Default TRUE.
cellsize	Cell size of the resulting heatmap. Default is 18.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
gtc <- hc_gene_to_cluster(hc)
f1 <- tempfile(fileext = ".tsv")
f2 <- tempfile(fileext = ".tsv")
utils::write.table(gtc, f1, sep = "\t", row.names = FALSE, quote = FALSE)
utils::write.table(gtc, f2, sep = "\t", row.names = FALSE, quote = FALSE)
hc <- hc_network_comparison_1(hc, gtc1_path = f1, gtc2_path = f2)
```

hc_network_comparison_2

Compare the edges of two networks for a set of genes

Description

Contrasts the neighbourhoods of gene_vec between two networks and reports which edges are shared and which are unique to either network.

Usage

```
hc_network_comparison_2(hc, net1, net2, as = "igraph", gene_vec)
```

Arguments

hc	A HCoCenaExperiment.
net1	First network as an igraph object.
net2	Second network as an igraph object.
as	Object type of net1/net2. Default is "igraph".
gene_vec	Character vector of genes to restrict the comparison to.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
g <- hc_graph(hc)
hc <- hc_network_comparison_2(
  hc,
  net1 = g,
  net2 = g,
  gene_vec = igraph::V(g)$name[1:3]
)
```

hc_pca	<i>pca</i>
--------	------------

Description

PCA plotting (S4 API)

Usage

```
hc_pca(hc, which = "all", color_by = NULL, ellipses = FALSE, cols = NULL)
```

Arguments

hc	A HCoCenaExperiment.
which	Which gene set to run the PCA on: "all" (default, all genes), "topvar" (the top-variable genes of each layer) or "network_genes" (only genes that ended up in the integrated network).
color_by	Optional annotation column name used to colour the samples. If NULL (default), the variable of interest from the global settings is used.
ellipses	Logical. Whether to draw group confidence ellipses. Default is FALSE.
cols	Optional named vector of colours for the groups. If NULL (default), a built-in palette is used.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("prepared")
hc <- hc_pca(hc)
```

hc_pca_algo_compare *pca algo compare*

Description

PCA algorithm comparison (S4 API)

Usage

```
hc_pca_algo_compare(hc, gtc = NULL, algo = NULL, cols = NULL)
```

Arguments

hc	A HCoCenaExperiment.
gtc	Optional gene-to-cluster table to compare against. If NULL (default), the current module assignment is used.
algo	Optional name of the clustering algorithm the gtc came from, used for the plot title.
cols	Optional named vector of colours for the sample groups.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_pca_algo_compare(hc)
```

hc_plot_celltype_annotation_matrix

Plot module x cell-type matrix from module cell-type annotation

Description

Uses `hc@satellite$celltype_annotation` and draws a heatmap-like matrix (modules x cell types) for easier interpretation than stacked barplots.

Usage

```
hc_plot_celltype_annotation_matrix(
  hc,
  database = NULL,
  value = c("pct", "count"),
  top_celltypes = 15,
  include_other = TRUE,
  other_label = "Other",
  cluster_order = NULL,
  low_color = "#f7fbff",
  high_color = "#08306b",
  return_data = TRUE
)
```

Arguments

<code>hc</code>	A HCoCenaExperiment.
<code>database</code>	Optional single database name. If NULL, uses combined selected_celltypes across databases.
<code>value</code>	One of "pct" or "count".
<code>top_celltypes</code>	Maximum number of cell types to show. Remaining cell types can be collapsed into other_label when include_other = TRUE.
<code>include_other</code>	Logical. Collapse non-top cell types into one column.
<code>other_label</code>	Label for the collapsed column.
<code>cluster_order</code>	Optional explicit module order (character vector).
<code>low_color</code>	Low-end fill color.
<code>high_color</code>	High-end fill color.
<code>return_data</code>	Logical. If TRUE, return a list with plot, matrix, and long_data; otherwise return only the plot.

Value

A ggplot object or a list containing plot + data.

Examples

```
hc <- hc_example_data("clustered")
gmt <- system.file("extdata", "toy_celltype_markers.gmt", package = "hcocena")
hc <- hc_celltype_annotation(
  hc,
  databases = character(0),
  custom_gmt_files = c(CellTypes = gmt),
  export_excel = FALSE
)
p <- hc_plot_celltype_annotation_matrix(hc)
```

hc_plot_cluster_heatmap
Plot module heatmap (S4 API)

Description

Plot module heatmap (S4 API)

Usage

```
hc_plot_cluster_heatmap(hc, file_name = "Heatmap_modules.pdf", ...)
```

Arguments

hc	A HCoCenaExperiment.
file_name	Optional export file name for the module heatmap. Defaults to "Heatmap_modules.pdf". Use FALSE to skip file export.
...	Additional plotting arguments forwarded to the heatmap backend, including smart_column_gaps, column_gap_by, and column_gap_mm.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_plot_cluster_heatmap(hc, file_name = FALSE)
```

hc_plot_cluster_heatmap_view
Plot a fixed-scale view of selected modules from the module heatmap

Description

Creates a module heatmap view for selected modules in the exact order supplied by modules. The module labels and module-color annotations are preserved, and the GFC expression color scale is inherited from the full heatmap cache instead of being recomputed from the subset.

Usage

```

hc_plot_cluster_heatmap_view(
  hc,
  modules,
  file_name = "Heatmap_modules_view.pdf",
  col_order = NULL,
  cluster_columns = FALSE,
  cluster_rows = FALSE,
  gfc_colors = NULL,
  gfc_scale_limits = NULL,
  preserve_full_heatmap = TRUE,
  store_view = TRUE,
  view_name = NULL,
  ...
)

```

Arguments

hc	A HCoCenaExperiment.
modules	Character/numeric vector of modules to show. Accepts current module labels such as "M2" or "M2.1", module colors, or numeric module indices in the current heatmap order.
file_name	Optional export file name for the view. Use FALSE to skip file export.
col_order	Optional condition order. If NULL, the stored full heatmap column order is reused when available.
cluster_columns	Logical. Whether to cluster columns in the view.
cluster_rows	Logical. Whether to cluster rows in the view. Defaults to FALSE so the modules vector controls the displayed order.
gfc_colors	Optional GFC palette override. If NULL, the stored full heatmap palette is reused.
gfc_scale_limits	Optional GFC scale override. If NULL, the stored full heatmap scale is reused; if unavailable, a full-cache/global fallback is used.
preserve_full_heatmap	Logical. If TRUE, restore the pre-existing full heatmap cache in hc after writing the view.
store_view	Logical. If TRUE, store lightweight metadata for the view under <code>hc@integration@cluster\$heatmap_view</code> .
view_name	Optional name used when <code>store_view = TRUE</code> .
...	Additional plotting arguments forwarded to <code>hc_plot_cluster_heatmap()</code> .

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_plot_cluster_heatmap_view(
  hc,
  modules = c("M2", "M3"),
  file_name = FALSE
)
```

hc_plot_cutoffs	<i>plot cutoffs</i>
-----------------	---------------------

Description

plot cutoffs

Usage

```
hc_plot_cutoffs(
  hc,
  interactive = TRUE,
  hline = list(R.squared = NULL, no_edges = NULL, no_nodes = NULL, no_networks = NULL)
)
```

Arguments

hc	A HCoCenaExperiment.
interactive	Logical. If TRUE (default), render the cutoff statistics as an interactive plotly widget; if FALSE, as a static plot.
hline	Named list of horizontal reference lines to draw, with the entries R.squared, no_edges, no_nodes and no_networks. NULL entries (the default) draw no line for that panel.

Value

Updated HCoCenaExperiment, invisibly; called for the cut-off diagnostic plot it draws.

Examples

```
hc <- hc_example_data("after_part1")
hc <- hc_plot_cutoffs(hc, interactive = FALSE)
```

hc_plot_deg_dist	<i>Plot degree distributions (S4 API)</i>
------------------	---

Description

Plots the logged degree distribution and its linear fit for each layer at the chosen correlation cutoff, to check how well the network follows a scale-free topology.

Usage

```
hc_plot_deg_dist(hc)
```

Arguments

hc	A HCoCenaExperiment.
----	----------------------

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("after_part1")
hc <- hc_plot_deg_dist(hc)
```

hc_plot_enrichment_panels	<i>Draw saved functional-enrichment panels from the current object</i>
---------------------------	--

Description

This is a convenience plotting helper for notebook/interactive usage. It draws the enrichment panels already stored in `hc@satellite$enrichments` after `hc_functional_enrichment()`, without recomputing enrichment.

Usage

```
hc_plot_enrichment_panels(hc, panels = NULL, heatmap_side = "left")
```

Arguments

hc	A HCoCenaExperiment.
panels	Optional character vector of panel names to draw. Default (NULL) draws all available panels in standard order. Accepted values include "top_Hallmark", "top_Kegg", "top_all_dbs", "top_all_dbs_mixed" (or without "top_" prefix).
heatmap_side	One of "left" (default) or "right".

Value

Invisibly returns hc.

Examples

```
hc <- hc_example_data("clustered")
gmt <- tempfile(fileext = ".gmt")
writeLines(
  c(
    paste(c("T cells", "-", "CD3D", "CD3E", "CD3G", "CD2", "CD28",
      "LCK", "ZAP70", "IL7R", "CD7", "TRAC"), collapse = "\t"),
    paste(c("B cells", "-", "CD19", "MS4A1", "CD79A", "CD79B", "BLNK",
      "PAX5", "CR2", "FCRL1", "TNFRSF13B", "VPREB3"), collapse = "\t"),
    paste(c("Monocytes", "-", "CD14", "LYZ", "FCN1", "VCAN", "S100A8",
      "S100A9", "CSF1R", "ITGAM", "CD68", "FCGR3A"), collapse = "\t"),
    paste(c("NK cells", "-", "NKG7", "GNLY", "KLRD1", "KLRF1", "PRF1",
      "GZMB", "NCR1", "KLRC1", "FGFBP2", "SPON2"), collapse = "\t")
  ),
  gmt
)
hc <- hc_functional_enrichment(
  hc,
  gene_sets = character(0),
  custom_gmt_files = c(CellTypes = gmt)
)
p <- hc_plot_enrichment_panels(hc)
```

hc_plot_enrichment_upstream_network

Plot module knowledge network from enrichment + upstream inference (S4 API)

Description

Plot module knowledge network from enrichment + upstream inference (S4 API)

Usage

```
hc_plot_enrichment_upstream_network(
  hc,
  enrichment_mode = "selected",
  upstream_mode = "selected",
  clusters = c("all"),
  max_enrichment_per_module = NULL,
  max_upstream_per_module = NULL,
  collapse_redundant_terms = TRUE,
  link_min_share = 0.25,
  label_mode = "both",
```

```

show_plot = TRUE,
save_pdf = TRUE,
pdf_name = "Module_Knowledge_Network.pdf",
gfc_scale_limits = NULL,
col_order = NULL,
heatmap_col_order = NULL,
cluster_columns = FALSE,
heatmap_cluster_columns = NULL,
pdf_width = NULL,
pdf_height = NULL,
pdf_pointsize = 11,
overall_plot_scale = 1
)

```

Arguments

hc	A HCoCenaExperiment.
enrichment_mode	Character scalar. One of "selected" (default) or "significant".
upstream_mode	Character scalar. One of "selected" (default) or "significant".
clusters	Either "all" (default) or a character vector of module colors to include.
max_enrichment_per_module	Optional positive integer. If set, keeps at most this many enrichment edges per module (best q-values first).
max_upstream_per_module	Optional positive integer. If set, keeps at most this many upstream edges per module (best q-values first).
collapse_redundant_terms	Logical. If TRUE (default), enrichment terms whose module genes lie mostly ($\geq 80\%$) in a better term of the same module are left out.
link_min_share	Minimum share of a regulator's targets in the module that must belong to a term for a term $\rightarrow$ regulator line; the term must also hold at least 1.5 times more of these targets than expected from its size in the module (at most 3 terms per regulator). Regulators without such a term are linked to the module directly. Default 0.25.
label_mode	Character scalar controlling term label density: "both" (default), "upstream_only", or "focus_only".
show_plot	Logical; if TRUE (default), prints the combined overview (heatmap + network) in the active graphics device.
save_pdf	Logical; if TRUE (default), writes a multi-page PDF to the current hCoCena save folder (overview + per-module focus pages).
pdf_name	Output PDF filename.
gfc_scale_limits	Optional numeric vector controlling the left module heatmap color scale limits. Provide one positive number ($x \rightarrow c(-x, x)$) or two numbers ($c(\min, \max)$). If NULL, uses upstream inference settings first (if available), then main heatmap settings, then $c(-\text{range_GFC}, \text{range_GFC})$.

col_order	Optional character vector overriding the hCoCena heatmap column order for this knowledge-network plot only. If NULL (default), the column order from the main module heatmap is reused when available.
heatmap_col_order	Legacy alias for col_order.
cluster_columns	Logical. If FALSE (default), reuse the column order from the main hCoCena heatmap when available. If TRUE, cluster the columns for this knowledge-network plot instead.
heatmap_cluster_columns	Legacy alias for cluster_columns.
pdf_width	Optional numeric width (inches) for network PDFs. If NULL (default), width is auto-estimated from content.
pdf_height	Optional numeric height (inches) for network PDFs. If NULL (default), height is auto-estimated from content.
pdf_pointsize	Numeric base pointsize used for PDF export devices. Default is 11.
overall_plot_scale	Numeric scaling factor for text and line sizes.

Value

Updated HCoCenaExperiment.

Examples

```
if (requireNamespace("decoupleR", quietly = TRUE)) {
  hc <- hc_example_data("clustered")
  gmt <- system.file("extdata", "toy_celltype_markers.gmt",
                     package = "hcocena")
  hc <- hc_functional_enrichment(
    hc,
    gene_sets = character(0),
    custom_gmt_files = c(CellTypes = gmt),
    universe = "network"
  )
  hc <- hc_upstream_inference(
    hc,
    resources = "Pathway",
    custom_pathway_gmt = c(CellTypes = gmt),
    plot = FALSE,
    save_pdf = FALSE
  )
  hc <- hc_plot_enrichment_upstream_network(hc, save_pdf = FALSE)
}
```

hc_plot_gfc_network	<i>plot gfc network</i>
---------------------	-------------------------

Description

Plot network colored by GFC (S4 API)

Usage

```
hc_plot_gfc_network(hc)
```

Arguments

hc	A HCoCenaExperiment.
----	----------------------

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_plot_gfc_network(hc)
```

hc_plot_integrated_network	<i>Plot integrated network (S4 API)</i>
----------------------------	---

Description

Plot integrated network (S4 API)

Usage

```
hc_plot_integrated_network(  
  hc,  
  layout = NULL,  
  layout_algorithm = NULL,  
  gene_labels = NULL,  
  save = TRUE,  
  store_plot = FALSE,  
  label_offset = 50  
)
```

Arguments

hc	A HCoCenaExperiment.
layout	Optional pre-computed layout matrix. If NULL (default), the stored layout is reused or a new one is computed.
layout_algorithm	Optional igraph layout function name, e.g. "layout_with_fr". Overrides the layout set in the global settings.
gene_labels	Optional character vector of genes to label in the plot.
save	Logical. Write the network to PDF. Default is TRUE.
store_plot	Logical. Keep the plot object in hc. Default is FALSE.
label_offset	Distance between a node and its label. Default is 50.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_plot_integrated_network(hc)
```

hc_plot_llm_enrichment

Plot LLM-based module interpretations

Description

Draws the interpretations stored by `hc_llm_enrichment()` or `hc_llm_import()`: a colored module box on the left and the inferred function on the right, optionally next to the module heatmap.

Usage

```
hc_plot_llm_enrichment(
  hc,
  slot_name = "llm_enrichment",
  modules = NULL,
  fields = c("general_processes", "contextual_state", "key_regulators",
    "grounded_processes"),
  max_chars = 90,
  text_size = 4,
  with_heatmap = TRUE,
  col_order = NULL,
  heatmap_col_order = NULL,
  cluster_columns = FALSE,
  heatmap_cluster_columns = NULL,
```

```

heatmap_rel_width = 1.7,
text_rel_width = 1.25,
title = "AI-assisted module interpretation",
module_label_fontsize = NULL,
module_label_pt_size = NULL,
module_box_width_cm = NULL,
save = TRUE,
file_stem = "LLM_enrichment",
pdf_width = NULL,
pdf_height = NULL,
dpi = 300
)

```

Arguments

hc	An HCoCenaExperiment with stored results from <code>hc_llm_enrichment()</code> or <code>hc_llm_import()</code> .
slot_name	Slot the results were stored under. Default is "llm_enrichment".
modules	Optional character vector to subset modules.
fields	Character vector selecting which fields to plot: "general_processes" (features only), "contextual_state" and "key_regulators" (with context) and "grounded_processes" (with context and evidence). Default: all four; "grounded_processes" is skipped when it was not requested.
max_chars	Approximate number of characters shown per term. Longer text is wrapped onto at most two lines (never inside parentheses); if it still does not fit, whole processes are dropped from the end, so the text ends at a " / ". Gene/protein lists in parentheses are left out of the plot. The heatmap cells stay square. Default is 90.
text_size	Numeric text size passed to <code>ggplot2::geom_text()</code> .
with_heatmap	Logical. If TRUE, reuse the stored hCoCena cluster heatmap and place the LLM interpretations as a right-side annotation. Falls back to a text-only plot if no heatmap cache is available.
col_order	Optional character vector overriding the hCoCena heatmap column order for this LLM plot only. If NULL (default), the column order from the main module heatmap is reused when available.
heatmap_col_order	Legacy alias for col_order.
cluster_columns	Logical. If FALSE (default), reuse the column order from the main hCoCena heatmap when available. If TRUE, cluster the columns for this LLM plot instead.
heatmap_cluster_columns	Legacy alias for cluster_columns.
heatmap_rel_width	Relative width of the heatmap panel when with_heatmap = TRUE.
text_rel_width	Relative width of the text panel when with_heatmap = TRUE.

title	Plot title. Default is "AI-assisted module interpretation".
module_label_fontsize	Optional numeric fontsize for module labels inside the colored module boxes. If NULL, a compact hc_plot_cluster_heatmap()-like default is used.
module_label_pt_size	Optional numeric glyph size for module labels inside the colored module boxes. Uses snpc units. If NULL, a compact hc_plot_cluster_heatmap()-like default is used.
module_box_width_cm	Optional numeric width (cm) for the module color boxes. If NULL, a compact hc_plot_cluster_heatmap()-like default is used.
save	Logical. If TRUE (default), export each selected field plot as both PDF and PNG into the configured hCoCena output directory.
file_stem	Base file name used for exported plots. Field names and, for single-module selections, the module label are appended automatically. Default is "LLM_enrichment".
pdf_width	Optional numeric export width in inches. If NULL, a field-appropriate default is chosen.
pdf_height	Optional numeric export height in inches. If NULL, a module-count dependent default is chosen.
dpi	Numeric export DPI for PNG output. Default is 300.

Value

A single plot object when one field is selected, otherwise a named list of plot objects.

Examples

```
# Interpretations imported from a web chat answer (no API key needed):
hc <- hc_example_data("clustered")
hc <- hc_plot_cluster_heatmap(hc, file_name = FALSE)
hc <- hc_llm_enrichment(hc, modules = "M1", provider = "manual",
                        verbose = FALSE)
id <- hc_satellite(hc, "llm_enrichment_pending")$id
answer <- paste0(
  '{"request_id": "', id, '", "results": [{ "set": "M1", ',
  '"general_processes": "B cell receptor signaling", ',
  '"contextual_state": "mature B cell program", ',
  '"key_regulators": "PAX5 / EBF1" } ]}'
)
hc <- hc_llm_import(hc, text = answer, verbose = FALSE)
p <- hc_plot_llm_enrichment(hc, with_heatmap = FALSE, save = FALSE)
```

`hc_plot_longitudinal_cap`
Plot CAP heatmap (donor x module-cluster proportions)

Description

Plot CAP heatmap (donor x module-cluster proportions)

Usage

```
hc_plot_longitudinal_cap(
  hc,
  slot_name = "longitudinal_endotypes",
  save_pdf = TRUE,
  file_prefix = "Longitudinal_CAP",
  show_values = FALSE,
  min_label_prop = 0.5,
  donor_order = NULL
)
```

Arguments

<code>hc</code>	A HCoCenaExperiment.
<code>slot_name</code>	Satellite slot with CAP outputs.
<code>save_pdf</code>	Logical. If TRUE, save PDF.
<code>file_prefix</code>	Prefix for output filename.
<code>show_values</code>	Logical. If TRUE, show proportions as text.
<code>min_label_prop</code>	Minimum proportion threshold for text labels.
<code>donor_order</code>	Optional donor order for x-axis. If NULL, donors are ordered by hierarchical clustering on the CAP matrix.

Value

List with `cap_heatmap` ggplot.

Examples

```
hc <- hc_example_data("clustered")
long <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
```

```

    impute = FALSE,
    seed = 1
  )
  hc <- long$hc
  p <- hc_plot_longitudinal_cap(hc, save_pdf = FALSE)

```

 hc_plot_longitudinal_endotype_meta_flow

Plot endotype-to-meta-cluster flow as Sankey diagram

Description

Plot endotype-to-meta-cluster flow as Sankey diagram

Usage

```

hc_plot_longitudinal_endotype_meta_flow(
  hc,
  slot_name = "longitudinal_endotypes",
  save_html = TRUE,
  file_prefix = "Longitudinal_Endotype_Meta_Flow"
)

```

Arguments

hc	A HCoCenaExperiment.
slot_name	Satellite slot with donor_cluster and meta_cluster.
save_html	Logical. If TRUE, save widget to HTML.
file_prefix	Output filename prefix.

Value

A list with sankey htmlwidget (or NULL if unavailable) and links table.

Examples

```

hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc

```

```

hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
  method = "kmeans",
  consensus = TRUE,
  seed = 1
)
p <- hc_plot_longitudinal_endotype_meta_flow(hc, save_html = FALSE)

```

hc_plot_longitudinal_endotypes

Plot longitudinal endotype results

Description

Creates a PCA donor plot and a module-wave plot (mean module value over time per endotype) from `hc_longitudinal_endotype_clustering()` outputs.

Usage

```

hc_plot_longitudinal_endotypes(
  hc,
  slot_name = "longitudinal_endotypes",
  save_pdf = TRUE,
  file_prefix = "Longitudinal_Endotypes",
  pca_width = 7,
  pca_height = 5,
  waves_width = 12,
  waves_height = 8
)

```

Arguments

<code>hc</code>	A HCoCenaExperiment.
<code>slot_name</code>	Satellite slot containing endotype clustering output.
<code>save_pdf</code>	Logical. If TRUE, save PDFs into the configured output folder.
<code>file_prefix</code>	Prefix for generated PDF names.
<code>pca_width, pca_height</code>	PDF size for PCA plot.
<code>waves_width, waves_height</code>	PDF size for trajectory plot.

Value

A named list with `pca` and `waves` ggplot objects.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_longitudinal_module_means(hc, donor_col = "donor", time_col = "timepoint")
hc <- hc_longitudinal_endotype_clustering(hc, k = 2)
p <- hc_plot_longitudinal_endotypes(hc, save_pdf = FALSE)
```

```
hc_plot_longitudinal_enrichment_meta_waves
```

Plot longitudinal enrichment waves by meta-cluster

Description

Creates one plot per enrichment database. For each module, the top x terms are placed into dedicated side-by-side panels (Top 1, Top 2, ...), and meta-cluster trajectories are drawn as longitudinal waves.

Usage

```
hc_plot_longitudinal_enrichment_meta_waves(
  hc,
  slot_name = "longitudinal_endotypes",
  databases = "Kegg",
  top = 3,
  custom_terms = NULL,
  term_match = c("exact", "contains", "regex"),
  enrichment_table = c("all", "selected", "significant"),
  score_method = c("ssgsea", "zscore", "rank_mean"),
  score_scale = c("none", "z"),
  layer = NULL,
  donor_col = NULL,
  time_col = NULL,
  time_levels = NULL,
  min_term_genes = 5,
  impute_missing = c("auto", "linear", "locf", "none"),
  qvalue_max = NULL,
  show_donor_lines = TRUE,
  donor_alpha = 0.04,
  donor_linewidth = 0.18,
  mean_linewidth = 2,
  mean_point_size = 1.8,
  show_ci = FALSE,
  ci_level = 0.95,
  ci_alpha = 0.12,
  free_y = TRUE,
  save_pdf = TRUE,
  file_prefix = "Longitudinal_Enrichment_MetaWaves",
  save_width = 12,
  save_height = 12,
```

```

    export_excel = TRUE
)

```

Arguments

<code>hc</code>	A HCoCenaExperiment.
<code>slot_name</code>	Satellite slot with <code>meta_cluster</code> results. Default: "longitudinal_endotypes". Use a concrete slot name, a shared prefix such as "longitudinal_endotypes" to process matching suffixed slots, or "all" to run over every compatible step 2 slot in <code>hc@satellite</code> .
<code>databases</code>	Character vector of databases to visualize (e.g. "Kegg").
<code>top</code>	Integer number of top terms per module.
<code>custom_terms</code>	Optional custom term filter (character vector or named module list). If set, matched terms are used instead of automatic top-term selection.
<code>term_match</code>	Matching mode for <code>custom_terms</code> : "exact", "contains", or "regex".
<code>enrichment_table</code>	Source enrichment table: "all" (default), "selected", or "significant".
<code>score_method</code>	"ssgsea" (default), "zscore", or "rank_mean".
<code>score_scale</code>	Plot-only scaling: "none" (default) or "z" to z-standardize scores within each module-term before plotting.
<code>layer</code>	Optional layer selector forwarded to <code>hc_plot_longitudinal_enrichment_waves()</code> .
<code>donor_col, time_col, time_levels</code>	Optional overrides forwarded to <code>hc_plot_longitudinal_enrichment_waves()</code> .
<code>min_term_genes</code>	Minimum matched genes per term.
<code>impute_missing</code>	Missing-value handling.
<code>qvalue_max</code>	Optional q-value cutoff before top-term selection.
<code>show_donor_lines</code>	Logical; include faint donor trajectories.
<code>donor_alpha, donor_linewidth</code>	Styling for donor lines.
<code>mean_linewidth, mean_point_size</code>	Styling for meta-cluster means.
<code>show_ci</code>	Logical; if TRUE, draw subtle confidence ribbons around meta-cluster mean trajectories.
<code>ci_level</code>	Confidence level in (0, 1) for ribbons (default 0.95).
<code>ci_alpha</code>	Ribbon alpha for confidence bands.
<code>free_y</code>	Logical; use free y scales per module row.
<code>save_pdf</code>	Logical; save one PDF per selected database.
<code>file_prefix</code>	Prefix for saved files.
<code>save_width, save_height</code>	PDF size.
<code>export_excel</code>	Logical; export tables per database.

Value

A list with plots, top_terms, donor_trajectories, mean_trajectories, and score_method_used. When multiple slots are processed, these top-level components are flattened with slot-prefixed names for convenient iteration, and nested per-slot outputs are also returned in results_by_slot, together with slot_info.

Examples

```
hc <- hc_example_data("clustered")
gmt <- system.file("extdata", "toy_celltype_markers.gmt", package = "hcocena")
hc <- hc_functional_enrichment(
  hc,
  gene_sets = character(0),
  custom_gmt_files = c(CellTypes = gmt),
  universe = "network"
)
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc
hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
  method = "kmeans",
  consensus = TRUE,
  seed = 1
)
waves <- hc_plot_longitudinal_enrichment_meta_waves(
  hc,
  databases = "Celltypes",
  top = 1,
  min_term_genes = 3,
  score_method = "rank_mean",
  save_pdf = FALSE,
  export_excel = FALSE
)
```

 hc_plot_longitudinal_enrichment_waves

Plot longitudinal enrichment waves for top terms per module

Description

Uses enrichment outputs from `hc_functional_enrichment()` to select top terms per module for one or multiple databases (for example Kegg), then computes sample-level term activity scores and visualizes trajectories across donor/timepoint in module-faceted wave plots.

Usage

```
hc_plot_longitudinal_enrichment_waves(
  hc,
  slot_name = "longitudinal_module_means",
  databases = "Kegg",
  top = 5,
  custom_terms = NULL,
  term_match = c("exact", "contains", "regex"),
  enrichment_table = c("selected", "all", "significant"),
  score_method = c("ssgsea", "zscore", "rank_mean"),
  score_scale = c("none", "z"),
  layer = NULL,
  donor_col = NULL,
  time_col = NULL,
  time_levels = NULL,
  min_term_genes = 5,
  impute_missing = c("auto", "linear", "locf", "none"),
  qvalue_max = NULL,
  show_donor_lines = TRUE,
  facet_ncol = 4,
  free_y = TRUE,
  save_pdf = TRUE,
  file_prefix = "Longitudinal_Enrichment_Waves",
  save_width = 13,
  save_height = 9,
  export_excel = TRUE
)
```

Arguments

<code>hc</code>	A HCoCenaExperiment.
<code>slot_name</code>	Satellite slot with longitudinal settings/module lookup. Usually "longitudinal_module_means".
<code>databases</code>	Character vector of databases to plot.
<code>top</code>	Number of top terms per module.
<code>custom_terms</code>	Optional custom term filter. Can be a character vector (applied to all modules) or a named list with module names (e.g. M1) as keys and character vectors of terms as values. If set, matched terms are plotted instead of automatic top-term selection.
<code>term_match</code>	Matching mode for custom_terms: "exact", "contains", or "regex".
<code>enrichment_table</code>	Which enrichment table to use: "selected" (default), "all", or "significant".

score_method	One of "ssgsea" (default), "zscore", or "rank_mean".
score_scale	Plot-only scaling: "none" (default) or "z" to z-standardize scores within each module-term before plotting.
layer	Optional layer selector (index/id/name). If NULL, uses layer_id from the longitudinal slot.
donor_col, time_col	Optional donor/time annotation columns. If NULL, values from the longitudinal slot are used.
time_levels	Optional explicit time ordering.
min_term_genes	Minimum number of matched genes per term.
impute_missing	One of "auto", "linear", "locf", "none".
qvalue_max	Optional q-value filter before top-term selection.
show_donor_lines	Logical; add faint donor lines behind term means.
facet_ncol	Number of facet columns.
free_y	Logical; free y-axis per module facet.
save_pdf	Logical; save one PDF per database.
file_prefix	Prefix used for output files.
save_width, save_height	PDF size in inches.
export_excel	Logical; export top-term + trajectory tables to Excel.

Details

Scoring uses GSVA ssGSEA when available (score_method = "ssgsea"), with a rank-based fallback if GSVA is unavailable.

Value

A list with plots, top_terms, mean_trajectories, donor_trajectories, and score_method_used.

Examples

```
hc <- hc_example_data("clustered")
gmt <- system.file("extdata", "toy_celltype_markers.gmt", package = "hccena")
hc <- hc_functional_enrichment(
  hc,
  gene_sets = character(),
  custom_gmt_files = c(CellTypes = gmt),
  universe = "network"
)
hc <- hc_longitudinal_module_means(hc, donor_col = "donor",
                                  time_col = "timepoint")
# Database names are title-cased by hc_functional_enrichment().
waves <- hc_plot_longitudinal_enrichment_waves(
  hc,
```

```

databases = "Celltypes",
top = 1,
min_term_genes = 3,
score_method = "rank_mean",
save_pdf = FALSE,
export_excel = FALSE
)
waves$top_terms

```

hc_plot_longitudinal_k_criterion

Plot k-criterion diagnostics for longitudinal clustering

Description

Plot k-criterion diagnostics for longitudinal clustering

Usage

```

hc_plot_longitudinal_k_criterion(
  hc,
  slot_name = "longitudinal_endotypes",
  save_pdf = TRUE,
  file_prefix = "Longitudinal_KCriterion",
  facet_ncol = 4,
  global_scope = c("auto", "meta", "endotype")
)

```

Arguments

hc	A HCoCenaExperiment.
slot_name	Satellite slot with longitudinal clustering outputs.
save_pdf	Logical. If TRUE, save PDFs.
file_prefix	Prefix for output files.
facet_ncol	Number of facet columns for module-wise diagnostics.
global_scope	One of "auto", "meta", "endotype". Controls which global k-selection table is shown.

Value

List with ggplots global_k and module_k.

Examples

```

hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc
hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
  method = "kmeans",
  consensus = TRUE,
  seed = 1
)
p <- hc_plot_longitudinal_k_criterion(hc, save_pdf = FALSE)

```

 hc_plot_longitudinal_meta_consensus

Plot consensus diagnostics for longitudinal meta-clustering

Description

Creates three consensus diagnostics:

1. donor-by-donor consensus matrix heatmap (selected k)
2. consensus metric trajectories across tested k (delta/PAC/coverage)
3. donor stability distribution per meta-cluster.

Usage

```

hc_plot_longitudinal_meta_consensus(
  hc,
  slot_name = "longitudinal_endotypes",
  save_pdf = TRUE,
  file_prefix = "Longitudinal_Meta_Consensus",
  show_donor_labels = FALSE,
  save_table = TRUE,
  table_format = c("xlsx", "csv")
)

```

Arguments

<code>hc</code>	A HCoCenaExperiment.
<code>slot_name</code>	Satellite slot with meta-clustering outputs.
<code>save_pdf</code>	Logical. If TRUE, save plots as PDFs.
<code>file_prefix</code>	Output filename prefix.
<code>show_donor_labels</code>	Logical. If TRUE, print donor IDs on consensus heatmap axes.
<code>save_table</code>	Logical. If TRUE, export consensus score table.
<code>table_format</code>	One of "xlsx" or "csv".

Value

A list with `consensus_heatmap`, optional `consensus_k`, optional `stability`, and optional `summary_table`.

Examples

```

hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc
hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
  method = "kmeans",
  consensus = TRUE,
  seed = 1
)
p <- hc_plot_longitudinal_meta_consensus(
  hc,
  save_pdf = FALSE,
  save_table = FALSE
)

```

 hc_plot_longitudinal_meta_embeddings

Plot meta-clustering embeddings and endotype cross-tab

Description

Plot meta-clustering embeddings and endotype cross-tab

Usage

```
hc_plot_longitudinal_meta_embeddings(
  hc,
  slot_name = "longitudinal_endotypes",
  save_pdf = TRUE,
  file_prefix = "Longitudinal_Meta",
  show_endotype_crosstab = FALSE,
  show_cluster_labels = TRUE,
  save_tables = TRUE,
  table_file_prefix = NULL,
  table_format = c("xlsx", "csv"),
  table_detail = c("simple", "full")
)
```

Arguments

hc	A HCoCenaExperiment.
slot_name	Satellite slot with meta-clustering outputs.
save_pdf	Logical. If TRUE, save plots as PDFs.
file_prefix	Output filename prefix.
show_endotype_crosstab	Logical. If TRUE, include an endotype-vs-meta-cluster overlap heatmap when endotype labels are present.
show_cluster_labels	Logical. If TRUE, label cluster centers in PCA and UMAP.
save_tables	Logical. If TRUE, export meta-clustering summary tables.
table_file_prefix	Optional prefix for summary tables. Defaults to file_prefix.
table_format	One of "xlsx" or "csv" for summary tables.
table_detail	One of "simple" or "full". "simple" exports only method-wise meta-cluster counts plus consensus decision summary.

Value

List with pca, optional umap, optional cross_tab, and tables.

Examples

```

hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)$hc
hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
  method = "kmeans",
  consensus = TRUE,
  seed = 1
)
p <- hc_plot_longitudinal_meta_embeddings(
  hc,
  save_pdf = FALSE,
  save_tables = FALSE
)

```

hc_plot_longitudinal_meta_module_waves

Plot module trajectories by donor meta-cluster

Description

Creates module-wise wave plots after meta-clustering: faint donor trajectories plus bold meta-cluster mean trajectories.

Usage

```

hc_plot_longitudinal_meta_module_waves(
  hc,
  slot_name = "longitudinal_endotypes",
  save_pdf = TRUE,
  file_prefix = "Longitudinal_Meta_ModuleWaves",
  donor_alpha = 0.13,
  donor_linewidth = 0.28,
  mean_linewidth = 1.2,
  mean_point_size = 1.9,
  facet_ncol = 4,

```

```

    free_y = TRUE,
    square_panels = TRUE,
    value_mode = c("scaled_mean_vst", "expression"),
    value_range = c(-2, 2),
    save_width = NULL,
    save_height = NULL
  )

```

Arguments

hc	A HCoCenaExperiment.
slot_name	Satellite slot with meta-clustering outputs.
save_pdf	Logical. If TRUE, save PDF.
file_prefix	Output filename prefix.
donor_alpha	Alpha for donor-level trajectories.
donor_linewidth	Line width for donor-level trajectories.
mean_linewidth	Line width for meta-cluster mean lines.
mean_point_size	Point size for meta-cluster mean points.
facet_ncol	Number of facet columns.
free_y	Logical. If TRUE, use free y-axis scales per module.
square_panels	Logical. If TRUE, use square facet panels.
value_mode	One of "scaled_mean_vst" or "expression". The default scales module values within each module before plotting.
value_range	Optional numeric vector of length 2 giving the y-axis range for value_mode = "scaled_mean_vst". Defaults to c(-2, 2). Use NULL to keep the automatic range.
save_width, save_height	Optional PDF width/height in inches.

Value

A list with one ggplot object (meta_module_waves).

Examples

```

hc <- hc_example_data("clustered")
hc <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,

```

```

    impute = FALSE,
    seed = 1
  )$hc
hc <- hc_longitudinal_endotype_clustering(hc, k = 2, nstart = 1, seed = 1)
hc <- hc_longitudinal_meta_clustering(
  hc,
  feature_source = "feature_matrix_used",
  k = 2:3,
  method = "kmeans",
  consensus = TRUE,
  seed = 1
)
p <- hc_plot_longitudinal_meta_module_waves(hc, save_pdf = FALSE)

```

hc_plot_longitudinal_module_clusters

Plot module-wise trajectory clusters and cluster heatmap

Description

Uses module-wise clustering results from [hc_longitudinal_endotype_clustering\(\)](#) and produces:

1. donor trajectories with module-cluster means per module
2. a module-by-donor cluster heatmap

Usage

```

hc_plot_longitudinal_module_clusters(
  hc,
  slot_name = "longitudinal_endotypes",
  save_pdf = TRUE,
  file_prefix = "Longitudinal_ModuleClusters",
  donor_alpha = 0.14,
  donor_linewidth = 0.28,
  mean_linewidth = 1.15,
  mean_point_size = 1.9,
  facet_ncol = 4,
  free_y = TRUE,
  show_heatmap_numbers = TRUE,
  square_panels = FALSE,
  save_waves_width = NULL,
  save_waves_height = NULL,
  save_heatmap_width = NULL,
  save_heatmap_height = NULL
)

```

Arguments

hc	A HCoCenaExperiment.
slot_name	Satellite slot containing results from hc_longitudinal_endotype_clustering().
save_pdf	Logical. If TRUE, save PDF outputs.
file_prefix	Prefix used for saved PDF file names.
donor_alpha	Alpha for donor trajectories.
donor_linewidth	Line width for donor trajectories.
mean_linewidth	Line width for cluster mean trajectories.
mean_point_size	Point size for cluster mean markers.
facet_ncol	Number of facet columns.
free_y	Logical. If TRUE, use free y-axis scales per module.
show_heatmap_numbers	Logical. If TRUE, overlay cluster IDs as text.
square_panels	Logical. If TRUE, force square panel aspect ratio for trajectory facets.
save_waves_width, save_waves_height	Optional PDF size for trajectory plot in inches.
save_heatmap_width, save_heatmap_height	Optional PDF size for CAP heatmap in inches.

Value

A list with ggplot objects: module_cluster_waves and module_cluster_heatmap.

Examples

```

hc <- hc_example_data("clustered")
long <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)
hc <- long$hc
p <- hc_plot_longitudinal_module_clusters(hc, save_pdf = FALSE)

```

hc_plot_longitudinal_module_means

Plot donor-level module trajectories (faint donors + mean line)

Description

Uses outputs from `hc_longitudinal_module_means()` to visualize donor trajectories per module. Individual donors are shown in a light stroke, with a highlighted module mean trajectory on top.

Usage

```
hc_plot_longitudinal_module_means(
  hc,
  slot_name = "longitudinal_module_means",
  save_pdf = TRUE,
  file_prefix = "Longitudinal_ModuleMeans",
  donor_alpha = 0.16,
  donor_linewidth = 0.35,
  mean_linewidth = 1.1,
  mean_point_size = 1.8,
  facet_ncol = 4,
  free_y = TRUE,
  square_panels = FALSE,
  save_width = NULL,
  save_height = NULL
)
```

Arguments

<code>hc</code>	A HCoCenaExperiment.
<code>slot_name</code>	Satellite slot containing results from <code>hc_longitudinal_module_means()</code> .
<code>save_pdf</code>	Logical. If TRUE, save PDF output.
<code>file_prefix</code>	Prefix used for saved PDF file names.
<code>donor_alpha</code>	Alpha for individual donor trajectories.
<code>donor_linewidth</code>	Line width for individual donor trajectories.
<code>mean_linewidth</code>	Line width for mean trajectories.
<code>mean_point_size</code>	Point size for mean trajectory markers.
<code>facet_ncol</code>	Number of facet columns.
<code>free_y</code>	Logical. If TRUE, use free y-axis scales per module.
<code>square_panels</code>	Logical. If TRUE, force square panel aspect ratio.
<code>save_width, save_height</code>	Optional PDF width/height in inches. If NULL, defaults are chosen automatically.

Value

A list with one ggplot object (module_means).

Examples

```
hc <- hc_example_data("clustered")
long <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)
hc <- long$hc
p <- hc_plot_longitudinal_module_means(hc, save_pdf = FALSE)
```

hc_plot_sample_distributions
plot sample distributions

Description

Plot sample distributions (S4 API)

Usage

```
hc_plot_sample_distributions(
  hc,
  plot_type = "boxplot",
  log_2 = TRUE,
  plot = TRUE
)
```

Arguments

hc	A HCoCenaExperiment.
plot_type	Either "boxplot" (default) or "freqdist" for a distribution-focused view.
log_2	Logical. Log2-transform the values before plotting. Default is TRUE.
plot	Logical. Draw the plot. Default is TRUE.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("prepared")
hc <- hc_plot_sample_distributions(hc)
```

```
hc_preview_celltype_database
```

Preview terms and genes from one Enrichr database

Description

Downloads and previews terms from a selected Enrichr library to help choose tissue/cell-type databases.

Usage

```
hc_preview_celltype_database(
  database,
  pattern = NULL,
  n = NULL,
  tissue_pattern = NULL,
  max_terms = 80,
  max_genes_per_term = 25,
  include_genes = TRUE
)
```

Arguments

database	Enrichr library name.
pattern	Backward-compatible alias for tissue_pattern.
n	Backward-compatible alias for max_terms.
tissue_pattern	Optional regular expression used to filter term names.
max_terms	Maximum number of terms to return. Default is 80.
max_genes_per_term	Maximum number of genes shown in genes_preview per term. Default is 25.
include_genes	Logical. If TRUE, include genes_preview.

Value

A data frame with terms and gene counts.

Examples

```
hc_preview_celltype_database("PanglaoDB")
```

`hc_print_longitudinal_endotypes`*Print longitudinal endotype outputs*

Description

Convenience printer for outputs from the stepwise longitudinal functions.

Usage

```
hc_print_longitudinal_endotypes(x, show_tables = TRUE)
```

Arguments

<code>x</code>	Output list from one of the longitudinal step functions, or a list containing nested <code>step1_module_donor</code> , <code>step2_meta_clustering</code> , and <code>step3_meta_module_trajectories</code> plot blocks.
<code>show_tables</code>	Logical. If TRUE, print optional meta-clustering tables when available.

Value

Invisibly returns `x`.

Examples

```
hc <- hc_example_data("clustered")
long <- hc_longitudinal_step1_module_donor(
  hc,
  donor_col = "donor",
  time_col = "timepoint",
  time_levels = c("T1", "T2"),
  k = 2,
  nstart = 1,
  cap_runs = 1,
  impute = FALSE,
  seed = 1
)
hc <- long$hc
hc_print_longitudinal_endotypes(long)
```

hc_read_data

*Function To Read All Count And Annotation Files***Description**

The function loads the count and annotation data for each layer and saves it in the hCoCena-Object's "data" slot. Genes with a variance of 0 will be automatically be excluded from the analysis.

Usage

```
hc_read_data(
  hc,
  sep_counts = "\t",
  sep_anno = "\t",
  gene_symbol_col = NULL,
  sample_col = NULL,
  count_has_rn = TRUE,
  anno_has_rn = TRUE,
  auto_setup_output = TRUE,
  create_output_dir = TRUE,
  project_folder = NULL
)
```

Arguments

hc	A HCoCenaExperiment.
sep_counts	The separator of the count files. Default is tab separated files. Ignore when loading data from objects instead of files.
sep_anno	The separator of the annotation files. Default is tab separated files. Ignore when loading data from objects instead of files.
gene_symbol_col	A String. Name of the column that contains the gene symbols. Ignore when loading data from objects instead of files.
sample_col	A String. Name of the column that contains the sample IDs. Ignore when loading data from objects instead of files.
count_has_rn	A Boolean. Whether or not the count file has rownames. Default is TRUE. Ignore when loading data from objects instead of files.
anno_has_rn	A Boolean. Whether or not the annotation file has rownames. Default is TRUE. Ignore when loading data from objects instead of files.
auto_setup_output	Boolean. If TRUE, run hc_check_dirs() and initialize the save folder before reading data (if dir_output is set).
create_output_dir	Boolean passed to hc_check_dirs() when auto_setup_output = TRUE.
project_folder	Optional save subfolder name. Use "" to write directly into dir_output. If NULL, keep an already configured save_folder or default to "".

Details

Count inputs are normalized to numeric expression matrices during import. For object-based count inputs, non-numeric helper columns are dropped. Genes with zero variance are removed during loading because they can destabilize downstream variance-based heuristics such as `hc_suggest_topvar()`. As a result, suggested inflection points can differ slightly from older releases on the same raw input.

Value

Updated HCoCenaExperiment.

Examples

```
extdir <- paste0(
  normalizePath(system.file("extdata", package = "hcocena"), winslash = "/"),
  "/"
)
outdir <- file.path(tempdir(), "hcocena-read-data")
dir.create(outdir, recursive = TRUE, showWarnings = FALSE)
hc <- hc_init()
hc <- hc_set_paths(
  hc,
  dir_count_data = extdir,
  dir_annotation = extdir,
  dir_reference_files = extdir,
  dir_output = outdir
)
hc <- hc_define_layers(
  hc,
  data_sets = list(
    Layer1 = c("toy_layer1_counts.tsv", "toy_layer1_anno.tsv"),
    Layer2 = c("toy_layer2_counts.tsv", "toy_layer2_anno.tsv")
  )
)
hc <- hc_read_data(
  hc,
  sep_counts = "\t",
  sep_anno = "\t",
  gene_symbol_col = "SYMBOL",
  sample_col = "SampleID",
  count_has_rn = FALSE,
  anno_has_rn = FALSE
)
names(MultiAssayExperiment::experiments(hc_mae(hc)))
```

hc_read_supplementary *Read Supplementary Data*

Description

Function to read and collect the supplementary files set in `set_supp_files()`.

Usage

```
hc_read_supplementary(hc)
```

Arguments

hc A HCoCenaExperiment.

Value

Updated HCoCenaExperiment.

Examples

```
refdir <- file.path(tempdir(), "hcocena-reference-files")
dir.create(refdir, recursive = TRUE, showWarnings = FALSE)
writeLines("ToySet\\tdescription\\tG1\\tG2", file.path(refdir, "hallmark.gmt"))
hc <- hc_init()
hc <- hc_set_paths(
  hc,
  dir_count_data = FALSE,
  dir_annotation = FALSE,
  dir_reference_files = paste0(normalizePath(refdir, winslash = "/"), "/"),
  dir_output = tempdir()
)
hc <- hc_set_supp_files(hc, Hallmark = "hallmark.gmt")
hc <- hc_read_supplementary(hc)
names(hcocena:::as_hcobject(hc)$supplementary_data)
```

hc_regroup_samples	<i>Regroup samples from module-level fold-change profiles</i>
--------------------	---

Description

Builds a module-by-sample fold-change matrix for one or more layers, clusters samples hierarchically, and cuts each dendrogram into k groups. By default, this is a non-destructive preview: the input object, its variable of interest, module assignments, and annotation files are not changed.

Usage

```
hc_regroup_samples(
  hc,
  layer = "all",
  modules = "all",
  k = 2,
  reference = c("mean", "control"),
  method = "complete",
  distance = "euclidean",
  group_col = "hc_regroup",
```

```

    apply = FALSE,
    overwrite = FALSE,
    show_values = FALSE,
    silent = !base::interactive()
  )

```

Arguments

hc	A HCoCenaExperiment with clustered modules.
layer	Layer selection. Use "all" (default), a layer index, layer id, configured layer name, or a vector of these values.
modules	Modules used as clustering features. Use "all" (default), current module labels such as "M2" or "M2.1", module colors, or numeric module indices.
k	Number of sample groups after cutting the dendrogram. Supply one integer for all selected layers or one integer per selected layer. Named vectors may use layer ids or configured layer names.
reference	Reference used for sample-level fold changes. "mean" compares each sample with the gene mean across all samples in the layer. "control" compares with the mean of samples matching the configured control label in the current variable-of-interest column.
method	Hierarchical clustering method passed to <code>stats::hclust()</code> .
distance	Distance used between samples. Supports methods accepted by <code>stats::dist()</code> and "correlation" for one minus Pearson correlation.
group_col	Name of the new annotation column created when <code>apply = TRUE</code> .
apply	Logical. If FALSE (default), return a preview without changing hc. If TRUE, add the inferred groups to group_col in each selected layer and store the diagnostics in <code>hc@satellite\$sample_regrouping</code> .
overwrite	Logical. Allow an existing group_col to be replaced when <code>apply = TRUE</code> . Defaults to FALSE.
show_values	Logical. Display rounded fold-change values in heatmap cells.
silent	Logical. If FALSE, draw each generated heatmap. The default is <code>!interactive()</code> so plots are shown in interactive R sessions.

Details

Fold changes are calculated for every gene in every sample relative to either that gene's mean across the layer or its mean in the configured control samples. Gene-level values are then averaged within the current modules. This follows the signed-ratio convention used by hCoCena: two-fold down is -2, no change is 1, and two-fold up is 2.

Value

A list with hc, per-layer results, plots, parameters, and applied. Each layer result contains sample_gfc, module_table, distance, tree, clusters, and plot.

Examples

```

hc <- hc_example_data("clustered")
preview <- hc_regroup_samples(hc, layer = 1, k = 2, silent = TRUE)
preview$results$set1$clusters
preview$results$set1$sample_gfc

applied <- hc_regroup_samples(
  hc,
  layer = "all",
  k = 2,
  group_col = "module_profile_group",
  apply = TRUE,
  silent = TRUE
)
hc <- applied$hc

```

 hc_resolve_paths

 Resolve hCoCena input/output directories for Docker or local projects

Description

Resolve hCoCena input/output directories for Docker or local projects

Usage

```

hc_resolve_paths(
  project_root_hint = NULL,
  local_count_subdir = "count_data",
  local_annotation_subdir = "annotation_data",
  local_reference_subdir = "reference_files",
  local_output_subdir = "output",
  docker_reference_dir = "/home/rstudio/reference_files",
  docker_count_dir = "/home/rstudio/count_data",
  docker_annotation_dir = "/home/rstudio/annotation_data",
  docker_output_dir = "/home/rstudio/output",
  create_docker_dirs = TRUE,
  fallback_count_data = "PATH_TO_FOLDER_THAT_HOLDS_YOUR_GENE_EXPRESSION_TABLES/",
  fallback_annotation = "PATH_TO_FOLDER_THAT_HOLDS_YOUR_ANNOTATION_TABLES/",
  fallback_reference_files = "PATH_TO_REFERENCE_FILES_FOLDER/",
  fallback_output = "PATH_TO_FOLDER_WHERE_HCOCENA_SHOULD_SAVE_ALL_ANALYSIS_OUTPUTS/"
)

```

Arguments

project_root_hint

Optional local project root to prioritize if it exists. If missing or not found, the current working directory is used.

```

local_count_subdir
    Local count-data subfolder below project_root_hint.
local_annotation_subdir
    Local annotation-data subfolder below project_root_hint.
local_reference_subdir
    Local reference-files subfolder below project_root_hint.
local_output_subdir
    Local output subfolder below project_root_hint.
docker_reference_dir
    Docker reference-files directory used to detect Docker mode.
docker_count_dir
    Docker count-data directory.
docker_annotation_dir
    Docker annotation-data directory.
docker_output_dir
    Docker output directory.
create_docker_dirs
    Boolean. If TRUE and Docker mode is detected, create the Docker directories
    when missing.
fallback_count_data
    Fallback count-data directory string used outside Docker when no local folder
    was detected.
fallback_annotation
    Fallback annotation directory string used outside Docker when no local folder
    was detected.
fallback_reference_files
    Fallback reference-files directory string used outside Docker when no local folder
    was detected.
fallback_output
    Fallback output directory string used outside Docker when no local folder was
    detected.

```

Value

Named list with `docker_mode`, `project_root`, `dir_count_data`, `dir_annotation`, `dir_reference_files`, and `dir_output`.

Examples

```

root <- file.path(tempdir(), "hcocena-paths")
dir.create(file.path(root, "reference_files"), recursive = TRUE, showWarnings = FALSE)
dir.create(file.path(root, "output"), recursive = TRUE, showWarnings = FALSE)
paths <- hc_resolve_paths(
  project_root_hint = root,
  local_count_subdir = NULL,
  local_annotation_subdir = NULL
)
names(paths)

```

 hc_run_expression_analysis_1

Run First Part Of The Gene Expression Analysis

Description

This function executes the first part of the data processing procedure. It leads up to choosing the correlation cut-off for each layer. All datasets will be filtered for their most variant genes as defined in the layer-specific settings. After this filtering step, the pair-wise correlation coefficients for all pairs of genes are calculated. Correlations that are negative or that have an associated p-value higher than 0.05 are immediately discarded. Next, a set of statistics will be calculated for the set range of cut-off values that aim to facilitate the cut-off choice. This includes determining the number of graph components resulting from creating a network when cutting the data with the respective cut-off, as well as the number of nodes and edges this network comprises. The last parameter that is evaluated is the R^2 -value of the data to a linear regression through the logged degree distribution for the given network.

Usage

```
hc_run_expression_analysis_1(
  hc,
  padj = "none",
  export = FALSE,
  import = NULL,
  bayes = FALSE,
  prior = 2,
  alpha = 0.5,
  corr_method = "pearson",
  corr_backend = "auto"
)
```

Arguments

hc	A HCoCenaExperiment.
padj	A String. Defines the method to be used for p-value adjustment. Valid values are "none" (default) or values for "method" in stats::p.adjust.
export	A Boolean. If TRUE, correlation values and p-values will be exported. This can save time if you plan on re-running the analysis since computing pair-wise correlations is a bottleneck of the analysis. Default is FALSE.
import	A list. Each slot in the list corresponds to one of the layers (datasets) and is either a vector of two strings (1. path to file holding the correlation matrix and 2. Path to the file holding the p-value matrix) or NA. A list slot is set to NA if for that layer you do not want to import a pre-calculated correlation matrix. The files do not necessarily have to be exported from a previous run, but can have any kind of origin (created with a different program or method). For compatibility it is only important, that it is a whitespace-separated text (.txt) file containing a

	symmetric, numeric matrix. The first line has to be gene names, there must be no row names, since the first line will be used for column names and row names. Also, you must provide a matrix with correlation values AND a matrix with corresponding p-values, where cells in the matrices correspond to each other (only a correlation matrix will not be sufficient). Default is NULL.
bayes	Sanchez-Taltavull et al. (2016) suggest superiority of Bayesian correlation analysis to Pearson correlation in some cases. Therefore, the Pearson correlation values can be weighted with Bayesian correlation values. To do so, set the "bayes"-parameter to TRUE. Default is FALSE, using only Pearson correlations.
prior	An integer, either 2 or 3, using prior 2 or 3 for the Bayes weighting as described in "Bayesian correlation analysis for sequence count data" by Sanchez-Taltavull et al. (2016).
alpha	A numeric value from 0 to 1. Allows to adjust the strength of the Bayes weighting: For alpha = 0 the Pearson correlation values remain unaltered, for alpha = 1 the Pearson correlation value and the Bayesian correlation value contribute equally to the final correlation.
corr_method	Correlation method to use. Supported values are "pearson" (default) and "spearman".
corr_backend	Correlation backend. "auto" (default) uses a fast cross-product path (BLAS matmul plus analytic t p-values) that is numerically identical to Hmisc::rcorr but ~13-25x faster, and automatically falls back to Hmisc::rcorr when the expression matrix contains NAs (to preserve pairwise-complete semantics). "rcorr" forces the original Hmisc::rcorr computation.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("prepared")
hc <- hc_run_expression_analysis_1(hc)
```

hc_run_expression_analysis_2

Run Second Part Of The Gene Expression Analysis

Description

This function plots a heatmap for the network genes in each data layer and computes the Group-Fold-Changes for all genes per layer.

Usage

```
hc_run_expression_analysis_2(
  hc,
  grouping_v = NULL,
  plot_HM = TRUE,
  method = "complete",
  additional_anno = NULL,
  cols = NULL
)
```

Arguments

hc	A HCoCenaExperiment.
grouping_v	A string giving a column name present in all annotation files, if this variable shall be used for grouping the samples instead of the variable of interest. Default is NULL.
plot_HM	A Boolean. Whether or not to plot the heatmap (for networks with many genes this may be very demanding for your computer if you are running the analysis locally). Default is TRUE.
method	The method used for clustering the heatmap in the pheatmap function. Default is "complete".
additional_anno	A list, with one slot per data set. A slot contains a vector of column names from that data set's annotation file that you wish to annotate with. If for some of the data sets you don't wish any further annotation, you can set the corresponding list slot to NULL. Default is NULL.
cols	A named list of color vectors. The list names need to match the chosen annotation column names. Default is NULL which uses implemented colors.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("after_part1")
hc <- hc_run_expression_analysis_2(hc, plot_HM = FALSE)
```

hc_satellite	<i>Access downstream results stored in an HCoCenaExperiment</i>
--------------	---

Description

Downstream steps (module gene lists, enrichments, hub genes, module statistics, ...) store their results in a named list inside the object. `hc_satellite()` returns that list, or a single entry of it.

Usage

```
hc_satellite(x, name = NULL)

## S4 method for signature 'HCoCenaExperiment'
hc_satellite(x, name = NULL)
```

Arguments

x An HCoCenaExperiment.

name Optional name of a single entry. If NULL (default), the whole list is returned.

Value

A `S4Vectors::SimpleList` of stored results, or the entry name (NULL if that entry does not exist).

Examples

```
hc <- hc_example_data("clustered")
names(hc_satellite(hc))
```

hc_set_cutoff	<i>Set network cutoffs (S4 API)</i>
---------------	-------------------------------------

Description

Set network cutoffs (S4 API)

Usage

```
hc_set_cutoff(
  hc,
  cutoff_vector = base::c(),
  auto = FALSE,
  fallback_cutoff = 0.982,
  verbose = TRUE
)
```

Arguments

hc A HCoCenaExperiment.

cutoff_vector A vector of correlation cutoff values, one per layer, in the order the layers were declared in [hc_define_layers\(\)](#). Ignored when `auto = TRUE`.

auto Logical; if TRUE, automatically select cutoffs from available tuning outputs in this priority:

1. `hc@satellite$cutoff_tuning$applied_cutoff_vector`,
2. `hc@satellite$cutoff_tuning$recommended_cutoff_vector`,

	3. hc@satellite\$auto_tune\$cutoff\$recommended_cutoff_vector, 4. internal simple auto cutoff extraction from layer results, 5. existing hc@config@layer\$cutoff, 6. user-provided cutoff_vector, 7. fallback_cutoff.
fallback_cutoff	Numeric fallback cutoff used for still-missing layers (for both auto = TRUE and auto = FALSE).
verbose	Logical; print additional source details. The final applied cutoff vector is always printed.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("after_part1")
hc <- hc_set_cutoff(hc, cutoff_vector = c(0.3, 0.3))
```

hc_set_global_settings

Define Global Settings

Description

Receives all settings that are globally valid, i.e., that are not dataset specific.

Usage

```
hc_set_global_settings(
  hc,
  organism = "human",
  control_keyword = "none",
  variable_of_interest = "merged",
  min_nodes_number_for_network = 50,
  min_nodes_number_for_cluster = 50,
  range_GFC = 2,
  layout_algorithm = "layout_with_stress",
  data_in_log = TRUE
)
```

Arguments

hc	A HCoCenaExperiment.
organism	Specification of the organism needed for all TF-related analyses. Set to "human" for human data and "mouse" for mouse data. Currently, hCoCena only supports human and mouse data, however, all analysis steps not related to TFs can be applied to other organisms.
control_keyword	Either 'none' if no controls are present (only possible when analysing one single dataset) or a string contained in the control sample descriptor of all annotation files, e.g. "healthy". The string must only be contained in the descriptor, e.g. "healthy" would work for "rhinovirusSetHealthy" and "influenzaSetHealthy", it does not have to match it perfectly.
variable_of_interest	The name of the column that must be present in all annotation files and which will be used for grouping samples, e.g., "condition".
min_nodes_number_for_network	An integer. The minimum number of nodes in the subsequently created network that can define a graph component. Graph components with less nodes will be discarded. Default is 50.
min_nodes_number_for_cluster	An integer. The minimum number of nodes that constitute a module/cluster when detecting community structures in the network. Default is 50.
range_GFC	A float. Defines the maximum value the group fold changes (GFCs) can acquire, all values above this value or beneath its negative will be truncated. Default is 2.0.
layout_algorithm	Layout algorithm used for the network. Supported values are: "layout_with_stress" (default), "layout_with_sparse_stress", "layout_with_fr", "layout_with_drl", "layout_with_kk" and "cytoscape". The stress-based layouts require the optional graphlayouts package and are typically more stable/readable than Fruchterman-Reingold for medium/large graphs. "cytoscape" uses an externally calculated Cytoscape layout.
data_in_log	Boolean. Whether or not the provided gene expression data is logged to the base of 2.

Value

Updated HCoCenaExperiment.

Examples

```
extdir <- paste0(
  normalizePath(system.file("extdata", package = "hcocena"), winslash = "/"),
  "/"
)
outdir <- file.path(tempdir(), "hcocena-global-settings")
dir.create(outdir, recursive = TRUE, showWarnings = FALSE)
```

```

hc <- hc_init()
hc <- hc_set_paths(
  hc,
  dir_count_data = extdir,
  dir_annotation = extdir,
  dir_reference_files = extdir,
  dir_output = outdir
)
hc <- hc_define_layers(
  hc,
  data_sets = list(
    Layer1 = c("toy_layer1_counts.tsv", "toy_layer1_anno.tsv"),
    Layer2 = c("toy_layer2_counts.tsv", "toy_layer2_anno.tsv")
  )
)
hc <- hc_read_data(
  hc,
  sep_counts = "\t",
  sep_anno = "\t",
  gene_symbol_col = "SYMBOL",
  sample_col = "SampleID",
  count_has_rn = FALSE,
  anno_has_rn = FALSE
)
hc <- hc_set_global_settings(
  hc,
  organism = "human",
  control_keyword = "control",
  variable_of_interest = "group",
  data_in_log = TRUE
)
as.data.frame(methods::slot(hc_config(hc), "global"))

```

hc_set_layer_settings *Define Layer Settings*

Description

Receives all settings that are dataset specific and not globally valid.

Usage

```

hc_set_layer_settings(
  hc,
  top_var,
  min_corr = 0.7,
  range_cutoff_length,
  print_distribution_plots = FALSE
)

```

Arguments

hc	A HCoCenaExperiment.
top_var	A vector with the length equal to the number of datasets/layers. Each entry of the vector is either "all" or an integer. Defines the number of most variable genes to be extracted per dataset. The order in the vector corresponds to the order in which the datasets have been declared in define_layers().
min_corr	A vector of floats with the length equal to the number of datasets/layers. To construct a meaningful co-expression network for each layer, correlation cut-offs must be determined for every dataset that mark the lower boundary for the correlation of two genes in order for their co-expression to be represented as an edge in the network. To facilitate choosing these cut-offs, a series of parameters will be calculated for a defined number of different cut-offs. The number of cut-offs for which these parameters are calculated is determined by 'range_cutoff_length'. The range from which these possible cut-offs are taken is on the lower end restricted by 'min_corr' and on the upper end by the maximum correlation calculated between any two genes in the dataset. Default is 0.7.
range_cutoff_length	A vector of integers with the length equal to the number of datasets/layers. Details see "min_corr".
print_distribution_plots	A vector of Booleans with the length equal to the number of datasets/layers. Whether or not to print the degree distribution plots for all tested cut-offs to pdf files. The number of plots per data set will therefore be equal to the 'range_cutoff_length' parameter you have set. Given the potential size, this should only be set to TRUE, if 'range_cutoff_length' is small or if one wishes to thoroughly analyse how the degree distribution changes in detail for differing cut-offs. Default is FALSE.

Value

Updated HCoCenaExperiment.

Examples

```
extdir <- paste0(
  normalizePath(system.file("extdata", package = "hcocena"), winslash = "/"),
  "/"
)
outdir <- file.path(tempdir(), "hcocena-layer-settings")
dir.create(outdir, recursive = TRUE, showWarnings = FALSE)
hc <- hc_init()
hc <- hc_set_paths(
  hc,
  dir_count_data = extdir,
  dir_annotation = extdir,
  dir_reference_files = extdir,
  dir_output = outdir
)
hc <- hc_define_layers(
```

```

    hc,
    data_sets = list(
      Layer1 = c("toy_layer1_counts.tsv", "toy_layer1_anno.tsv"),
      Layer2 = c("toy_layer2_counts.tsv", "toy_layer2_anno.tsv")
    )
  )
  hc <- hc_read_data(
    hc,
    sep_counts = "\t",
    sep_anno = "\t",
    gene_symbol_col = "SYMBOL",
    sample_col = "SampleID",
    count_has_rn = FALSE,
    anno_has_rn = FALSE
  )
  hc <- hc_set_layer_settings(
    hc,
    top_var = c(2, 2),
    min_corr = 0.1,
    range_cutoff_length = c(2, 2),
    print_distribution_plots = FALSE
  )
  as.data.frame(methods::slot(hc_config(hc), "layer"))

```

hc_set_paths

Set input/output paths in an HCoCenaExperiment

Description

Set input/output paths in an HCoCenaExperiment

Usage

```

hc_set_paths(
  hc,
  dir_count_data,
  dir_annotation,
  dir_reference_files,
  dir_output
)

```

Arguments

hc A HCoCenaExperiment.

dir_count_data Path to count data.

dir_annotation Path to annotation data.

dir_reference_files Path to reference files.

dir_output Output directory.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_init()
hc <- hc_set_paths(
  hc,
  dir_count_data = FALSE,
  dir_annotation = FALSE,
  dir_reference_files = tempdir(),
  dir_output = tempdir()
)
as.data.frame(methods::slot(hc_config(hc), "paths"))
```

hc_set_supp_files	<i>Set Supplementary Files</i>
-------------------	--------------------------------

Description

Receives the file names of the supplementary files

Usage

```
hc_set_supp_files(
  hc,
  Tf = NULL,
  Hallmark = NULL,
  Go = NULL,
  Kegg = NULL,
  Reactome = NULL,
  ...
)
```

Arguments

hc	A HCoCenaExperiment.
Tf	A file name. The file must contain a list of at least two columns. One column should be titled with the organism from which the datasets originate and contain gene names, the other column must always be the last column and contain the information if the gene is a transcription factor ("TF"), a co-factor ("Co_factor"), a chromatin remodelling protein ("Chromatin_remodeller") or a ribonucleic acid binding protein ("RNBP"). The name of the last column may vary. An exemplary file for mouse and human is found in the Reference Files Folder in the repository.
Hallmark	A file name for a .gmt MSigDB hallmark gene set file. You can find one in the reference file folder in the repository.

Go	A file name for a .gmt gene ontology file. You can find one in the reference file folder in the repository.
Kegg	A file name for a .gmt KEGG database file. You can find one in the reference file folder in the repository.
Reactome	A file name for a .gmt Reactome database file. You can find one in the reference file folder in the repository.
...	Additional supplementary files passed through to set_supp_files().

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_init()
hc <- hc_set_supp_files(
  hc,
  Hallmark = "hallmark.gmt",
  Go = "go.gmt"
)
sort(names(hcocena:::as_hcobject(hc)$supplement))
```

hc_split_modules	<i>Split one or multiple modules into submodules (S4 API)</i>
------------------	---

Description

Re-clusters genes within selected modules and replaces those modules by submodules (with color shades of the parent module).

Usage

```
hc_split_modules(
  hc,
  modules,
  cluster_algo = "cluster_leiden",
  no_of_iterations = 2,
  resolution = 0.1,
  resolution_grid = NULL,
  resolution_test_only = FALSE,
  partition_type = "RBConfigurationVertexPartition",
  seed = 168575,
  drop_small_submodules = TRUE,
  min_submodule_size = NULL,
  min_module_size = NULL,
  verbose = TRUE
)
```

Arguments

hc	A HCoCenaExperiment.
modules	Character/numeric vector of modules to split. Accepts module labels (e.g. "M3"), module colors, or module indices.
cluster_algo	Clustering algorithm for within-module splitting. One of "cluster_leiden" (default), "cluster_louvain", "cluster_fast_greedy", "cluster_infomap", "cluster_walktrap", "cluster_label_prop" or "auto".
no_of_iterations	Number of Leiden iterations (used only for Leiden).
resolution	Leiden resolution (used only for Leiden). Use either one positive value for all selected modules or one positive value per selected module in the same order as modules. Named vectors may use module labels or module colors.
resolution_grid	Optional numeric vector of candidate resolutions to test before splitting. For each candidate, hCoCena reports how many submodules would be retained after size filtering.
resolution_test_only	Logical; if TRUE, only run the resolution test (when resolution_grid is set) and do not apply any split.
partition_type	Leiden partition type (used only for Leiden).
seed	Random seed used for deterministic clustering.
drop_small_submodules	Logical; if TRUE (default), submodules with fewer than min_submodule_size genes are dropped from the module annotation (their genes become unassigned/white in network visualizations).
min_submodule_size	Optional minimum size for retained split submodules. If NULL, uses global_settings\$min_nodes_number.
min_module_size	Optional alias for min_submodule_size. If set, it takes precedence.
verbose	Logical; print progress messages.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_plot_cluster_heatmap(hc, file_name = FALSE)
hc <- hc_split_modules(hc, modules = "M1", resolution = 1)
```

hc_suggest_topvar	<i>suggest topvar</i>
-------------------	-----------------------

Description

Suggest top variable genes (S4 API)

Usage

```
hc_suggest_topvar(hc)
```

Arguments

hc A HCoCenaExperiment.

Details

hc_suggest_topvar() operates on the counts stored in hc, i.e. after hc_read_data() preprocessing. Because hc_read_data() removes zero-variance genes and drops non-numeric helper columns from object-based count inputs, suggested inflection points can differ slightly from older releases on the same raw input even though the hc_suggest_topvar() heuristic itself is unchanged.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("prepared")
hc <- hc_suggest_topvar(hc)
```

hc_tf_overrep_module	<i>TF enrichment per module (S4 API)</i>
----------------------	--

Description

TF enrichment per module (S4 API)

Usage

```
hc_tf_overrep_module(hc, clusters = "all", topTF = 5, topTarget = 5)
```

Arguments

hc	A HCoCenaExperiment.
clusters	Either "all" (default) or a vector of module colours/labels for which the TF enrichment should be performed.
topTF	Integer. Number of top ranking transcription factors to return per module. Default is 5.
topTarget	Integer. Number of top ranking targets to return per transcription factor. Default is 5.

Value

Updated HCoCenaExperiment.

Examples

```
# Queries the ChEA3 web service.  
hc <- hc_example_data("clustered")  
hc <- hc_tf_overrep_module(hc, topTF = 3, topTarget = 3)
```

hc_tf_overrep_network *TF enrichment network-wide (S4 API)*

Description

TF enrichment network-wide (S4 API)

Usage

```
hc_tf_overrep_network(hc, topTF = 100, topTarget = 30)
```

Arguments

hc	A HCoCenaExperiment.
topTF	Integer. Number of transcription factors with the highest number of enriched targets network-wide. Default is 100.
topTarget	Integer. Number of top enriched targets to return per transcription factor. Default is 30.

Value

Updated HCoCenaExperiment.

Examples

```
# Queries the ChEA3 web service.
hc <- hc_example_data("clustered")
hc <- hc_tf_overrep_network(hc, topTF = 10, topTarget = 5)
names(hc_satellite(hc, "tf_network_targets"))
```

hc_tune_cutoff

Robust tiered correlation-cutoff tuning

Description

Short-name wrapper around `hc_auto_tune_cutoff_tiered()` for daily use. The report is stored in `hc@satellite$cutoff_tuning`.

Usage

```
hc_tune_cutoff(
  hc,
  apply = FALSE,
  tier1_sft_min = 0.85,
  tier1_mean_k_min = 10,
  tier2_sft_min = 0.8,
  tier2_mean_k_min = 5,
  fallback_sft_min = 0.7,
  require_negative_slope = TRUE,
  max_no_networks = NULL,
  max_components = NULL,
  min_node_fraction = NULL,
  fallback_respect_base_filters = TRUE,
  verbose = TRUE
)
```

Arguments

hc	A HCoCenaExperiment object.
apply	Logical; if TRUE, apply the resulting cutoff vector with <code>hc_set_cutoff()</code> . Missing tiered recommendations fall back to existing cutoffs from <code>hc@config@layer\$cutoff</code> where possible.
tier1_sft_min	Numeric minimum SFT fit (R.squared) for the strict rule.
tier1_mean_k_min	Numeric minimum mean connectivity (mean_k) for the strict rule.
tier2_sft_min	Numeric minimum SFT fit (R.squared) for the relaxed rule.
tier2_mean_k_min	Numeric minimum mean connectivity (mean_k) for the relaxed rule.

fallback_sft_min	Numeric lower bound for best_available selection using the maximum available SFT fit.
require_negative_slope	Logical; require a negative slope ($\text{slope} < 0$) in Tier 1/Tier 2 (and optionally fallback, see fallback_respect_base_filters).
max_no_networks	Optional numeric scalar. If set, only cutoffs with <code>no_of_networks</code> $\leq$ <code>max_no_networks</code> are accepted in Tier 1/Tier 2.
max_components	Deprecated alias for <code>max_no_networks</code> (kept for backward compatibility).
min_node_fraction	Optional numeric in $(0, 1]$. If set, require <code>no_nodes</code> $\geq$ <code>min_node_fraction</code> * <code>max(no_nodes)</code> in Tier 1/Tier 2.
fallback_respect_base_filters	Logical; if TRUE, best_available selection is restricted by base filters (<code>slope</code> , <code>max_no_networks</code> , <code>min_node_fraction</code>).
verbose	Logical; print per-layer recommendation messages.

Details

Key tuning knobs:

- `tier1_sft_min`, `tier1_mean_k_min`: strict-tier thresholds.
- `tier2_sft_min`, `tier2_mean_k_min`: relaxed-tier thresholds.
- `fallback_sft_min`: minimum fit for best_available selection.
- `require_negative_slope`: enforce negative slope in tier filters.
- `max_no_networks`: optional upper bound for connected-component count.
- `min_node_fraction`: optional retained-node requirement relative to best available node count across tested cutoffs.
- `fallback_respect_base_filters`: whether best_available must respect the same base filters (`slope`/components/nodes).

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("after_part1")
hc <- hc_tune_cutoff(hc, apply = FALSE, verbose = FALSE)
```

hc_tune_resolution	<i>Tune clustering algorithm and resolution</i>
--------------------	---

Description

Convenience wrapper for clustering-focused auto-tuning. Internally this calls `hc_auto_tune(..., tune_cutoff = FALSE, tune_clustering = TRUE)`. The clustering report is additionally mirrored to `hc@satellite$resolution_tuning`.

Usage

```
hc_tune_resolution(
  hc,
  apply = FALSE,
  cluster_algorithms = c("cluster_leiden", "cluster_louvain"),
  resolution_grid = base::seq(0.05, 0.5, by = 0.05),
  no_of_iterations = 2,
  partition_type = "RBConfigurationVertexPartition",
  max_cluster_count_per_gene = 1,
  module_count_bounds = c(3, 60),
  prefer_resolution = 0.1,
  verbose = TRUE
)
```

Arguments

hc	A HCoCenaExperiment object.
apply	Logical; if TRUE, apply the best clustering recommendation.
cluster_algorithms	Character vector of candidate algorithms.
resolution_grid	Numeric vector of Leiden resolutions.
no_of_iterations	Passed to clustering.
partition_type	Passed to clustering. For Leiden, resolution sweeps are only informative with "CPMVertexPartition" or "RBConfigurationVertexPartition".
max_cluster_count_per_gene	Passed to clustering.
module_count_bounds	Optional length-2 numeric vector (min_modules, max_modules) used as a soft constraint.
prefer_resolution	Tie-break preference for resolution.
verbose	Logical; print progress messages.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_tune_resolution(hc, apply = FALSE)
```

hc_unsplit_modules	<i>Undo module splitting (S4 API)</i>
--------------------	---------------------------------------

Description

Restores module structure from split history.

Usage

```
hc_unsplit_modules(hc, which = c("last", "all"), verbose = TRUE)
```

Arguments

hc	A HCoCenaExperiment.
which	Either "last" (undo one split step) or "all" (restore the original pre-split cluster state).
verbose	Logical; print progress messages.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_plot_cluster_heatmap(hc, file_name = FALSE)
hc <- hc_split_modules(hc, modules = "M1", resolution = 1)
hc <- hc_unsplit_modules(hc, which = "all")
```

```
hc_update_clustering_algorithm
      update clustering algorithm
```

Description

Update clustering algorithm (S4 API)

Usage

```
hc_update_clustering_algorithm(hc, new_algo = NULL, gtc = NULL)
```

Arguments

hc	A HCoCenaExperiment.
new_algo	Name of the clustering algorithm to switch to, e.g. "cluster_louvain". Mutually exclusive with gtc.
gtc	Optional externally supplied gene-to-cluster table to adopt instead of re-running an algorithm.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_update_clustering_algorithm(hc, new_algo = "cluster_louvain")
```

```
hc_upstream_inference  Upstream regulator/pathway inference (S4 API)
```

Description

Links each module to transcription factors and signalling pathways whose target genes are over-represented among the module genes.

Usage

```
hc_upstream_inference(
  hc,
  resources = c("TF", "Pathway"),
  top = 5,
  clusters = c("all"),
  padj = "BH",
```

```

qval = 0.05,
tf_resource = "auto",
tf_confidence = c("A", "B", "C"),
progeny_top = 100,
minsize = 5,
min_overlap = 3,
collapse_redundant = TRUE,
redundancy_jaccard = 0.5,
method = "ora",
activity_input = "gfc",
fc_comparisons = NULL,
custom_pathway_gmt = NULL,
heatmap_side = "left",
cluster_columns = FALSE,
heatmap_cluster_columns = NULL,
col_order = NULL,
heatmap_col_order = NULL,
gfc_scale_limits = NULL,
plot = TRUE,
save_pdf = TRUE,
pdf_width = NULL,
pdf_height = NULL,
pdf_pointsize = 11,
plot_per_comparison = TRUE,
consistent_terms = TRUE,
overall_plot_scale = 1
)

```

Arguments

hc	A HCoCenaExperiment.
resources	Character vector of upstream resources to use. Allowed values are "TF" and "Pathway". Default is both.
top	Integer. Number of top significant regulators/pathways to keep per module and resource in the selected summary. Default is 5.
clusters	Either "all" (default) or a character vector of module colors to process.
padj	Multiple-testing correction method passed to <code>stats::p.adjust()</code> . Default is "BH".
qval	Adjusted p-value threshold for significance. Default is 0.05.
tf_resource	TF regulon resource: "auto" (default) uses CollecTRI when it can be loaded (needs the OmnipathR package and internet) and DoRothEA from the dorothea package otherwise; "collectri" or "dorothea" force one of them. The resource used is stored in <code>settings\$tf_database</code> and in the database column.
tf_confidence	Character vector of DoRothEA confidence levels to keep (DoRothEA only). Default is c("A", "B", "C").
progeny_top	Number of most responsive genes per PROGENy pathway (smallest p-value in the PROGENy model). Default is 100.

<code>minsize</code>	Minimum number of targets among the network genes a regulator needs to be tested; also the minimum module size. Default is 5.
<code>min_overlap</code>	Minimum number of a regulator's targets in a module for the link to count as significant. Default is 3.
<code>collapse_redundant</code>	Logical. If TRUE (default), regulators whose module targets largely repeat a better-ranked regulator of the same module (see <code>redundancy_jaccard</code>) are named in the column <code>redundant_with</code> and left out of the selected (plotted) regulators; they stay in the tables.
<code>redundancy_jaccard</code>	Jaccard index of the module targets from which a regulator counts as redundant. Default is 0.5.
<code>method</code>	Test that links modules and regulators. Only "ora" (hypergeometric over-representation, default) is supported.
<code>activity_input</code>	Character scalar selecting the values used for the signed score per condition: "gfc" (default) uses <code>integrated_output\$GFC_all_layers</code> , "fc" uses user-defined pairwise log2 fold-changes of the group means from <code>fc_comparisons</code> .
<code>fc_comparisons</code>	Character vector of pairwise comparisons used only when <code>activity_input</code> = "fc". Each entry must be formatted as "groupA_vs_groupB" (numerator vs denominator), e.g. <code>c("IFNg_seq_vs_baseline_seq", "IL4_seq_vs_baseline_seq")</code> .
<code>custom_pathway_gmt</code>	Optional custom pathway GMT file(s) added to the pathway inference resource. Accepts a character vector or named list of file paths. Paths can be absolute/relative or file names inside <code>dir_reference_files</code> .
<code>heatmap_side</code>	Position of the hCoCena heatmap in the combined output. Choose one of "left" (default) or "right".
<code>cluster_columns</code>	Logical. If FALSE (default), reuse the column order from the main hCoCena heatmap when available. If TRUE, cluster the columns for this upstream plot instead.
<code>heatmap_cluster_columns</code>	Legacy alias for <code>cluster_columns</code> .
<code>col_order</code>	Optional character vector overriding the hCoCena heatmap column order for this upstream plot only. If NULL (default), the column order from the main module heatmap is reused when available.
<code>heatmap_col_order</code>	Legacy alias for <code>col_order</code> .
<code>gfc_scale_limits</code>	Optional numeric vector controlling the module-heatmap color scale limits used in upstream combined heatmaps (left/right hCoCena panel). Provide one positive number (<code>x -> c(-x, x)</code>) or two numbers (<code>c(min, max)</code>). If NULL, uses stored limits from the main heatmap when available, otherwise falls back to <code>c(-range_GFC, range_GFC)</code> .
<code>plot</code>	Logical; if TRUE (default), draw plot outputs in the active graphics device.
<code>save_pdf</code>	Logical; if TRUE (default), export plots to <code>Upstream_Inference.pdf</code> .

pdf_width	Optional numeric width (inches) for Upstream_Inference.pdf. If NULL (default), width is auto-estimated from content.
pdf_height	Optional numeric height (inches) for Upstream_Inference.pdf. If NULL (default), height is auto-estimated from content.
pdf_pointsize	Numeric base pointsize used for the upstream PDF device. Default is 11.
plot_per_comparison	Logical; if TRUE, additionally create one combined upstream heatmap page per activity column (GFC condition or FC comparison), each with the matching one-column module heatmap.
consistent_terms	Logical; controls term comparability when plot_per_comparison = TRUE. If TRUE, per-comparison pages still show only values from the currently shown condition, but use a global (all-condition) term axis for comparability; * marks regulators linked to the module and active in the currently shown condition. If FALSE, each page uses only local selected activities from the shown condition and no significance marker is drawn.
overall_plot_scale	Numeric scaling factor for plot typography and marker sizes. Default is 1.

Details

For every module and regulator, a hypergeometric test compares the regulator's targets in the module with its targets among all network genes (the genes that could have landed in a module). P-values are adjusted over all module x regulator tests of a resource. TF regulons come from CollecTRI (or DoRothEA), pathway-responsive genes from PROGENy (top progeny_top genes per pathway) and optional custom GMT files.

A link is significant when its adjusted p-value (qvalue) is at most qval and at least min_overlap targets lie in the module.

For each link the result also reports

- score: how the module's targets of the regulator move, as the mean of mode of regulation x GFC (or log2 FC) per condition; positive values mean the targets change as expected for a more active regulator. The overall score is the one of the condition with the largest absolute value (peak_condition).
- consistency: share of those targets that move in the direction of the score.
- activity, activity_qvalue (per condition): regulator activity estimated with decoupleR: :run_ulm() over all network genes, adjusted per condition. A regulator counts as active in a condition when this activity is significant and has the same sign as the module score; n_active_conditions counts those conditions, and the per-condition heatmaps mark them with *.
- regulator_module, regulator_in_module, regulator_cor (TFs): the module of the TF gene itself and the correlation of its profile with the module mean. A TF co-expressed with its targets is framed in the heatmaps.
- redundant_with: a better-ranked regulator of the same module whose module targets largely overlap (see collapse_redundant).

Value

Updated HCoCenaExperiment.

Examples

```
if (requireNamespace("decoupleR", quietly = TRUE)) {
  hc <- hc_example_data("clustered")
  gmt <- system.file("extdata", "toy_celltype_markers.gmt",
    package = "hccocena")
  # Pathways from a local GMT; TF regulons (`resources = "TF"`) need
  # the `OmnipathR` (CollectRI) or `dorothea` package.
  hc <- hc_upstream_inference(
    hc,
    resources = "Pathway",
    custom_pathway_gmt = c(CellTypes = gmt),
    plot = FALSE,
    save_pdf = FALSE
  )
  head(hc_satellite(hc, "upstream_inference")$selected_upstream_all)
}
```

hc_user_specific_cluster_profiling

Profile modules against user-defined enrichment terms

Description

Scores every module against a user-supplied list of terms of interest, either given directly via `enrichment_keys` or read from a file.

Usage

```
hc_user_specific_cluster_profiling(
  hc,
  from_file = FALSE,
  path = NULL,
  enrichment_keys = NULL,
  db = c("Go", "Kegg", "Hallmark"),
  padj = "BH",
  qval = 0.1
)
```

Arguments

hc	A HCoCenaExperiment.
from_file	Logical. If TRUE, read the terms of interest from path instead of enrichment_keys. Default is FALSE.

path	Path to the file holding the terms of interest. Only used when from_file = TRUE.
enrichment_keys	Character vector of terms to profile the modules against. Required when from_file = FALSE.
db	Databases to search for the terms. Default is c("Go", "Kegg", "Hallmark").
padj	Multiple-testing correction method. Default is "BH".
qval	Maximum adjusted p-value for a term to count. Default is 0.1.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
gmt <- tempfile(fileext = ".gmt")
writeLines(
  c(
    paste(c("T cells", "-", "CD3D", "CD3E", "CD3G", "CD2", "CD28",
            "LCK", "ZAP70", "IL7R", "CD7", "TRAC"), collapse = "\t"),
    paste(c("B cells", "-", "CD19", "MS4A1", "CD79A", "CD79B", "BLNK",
            "PAX5", "CR2", "FCRL1", "TNFRSF13B", "VPREB3"), collapse = "\t"),
    paste(c("Monocytes", "-", "CD14", "LYZ", "FCN1", "VCAN", "S100A8",
            "S100A9", "CSF1R", "ITGAM", "CD68", "FCGR3A"), collapse = "\t"),
    paste(c("NK cells", "-", "NKG7", "GNLY", "KLRD1", "KLRF1", "PRF1",
            "GZMB", "NCR1", "KLRC1", "FGFBP2", "SPON2"), collapse = "\t")
  ),
  gmt
)
hc <- hc_functional_enrichment(
  hc,
  gene_sets = character(),
  custom_gmt_files = c(CellTypes = gmt)
)
hc <- hc_user_specific_cluster_profiling(hc, from_file = FALSE)
```

hc_visualize_gene_expression

Visualize gene expression (S4 API)

Description

Plots the mean expression per condition for a set of genes as a heatmap.

Usage

```
hc_visualize_gene_expression(
  hc,
  genes,
  name = NULL,
  width = 15,
  height = 10,
  save = TRUE
)
```

Arguments

hc	A HCoCenaExperiment.
genes	Character vector of gene symbols to plot.
name	Optional name used for the plot title and output file. If NULL (default), a name is derived automatically.
width	Plot width in inches. Default is 15.
height	Plot height in inches. Default is 10.
save	Logical. Write the heatmap to PDF. Default is TRUE.

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("clustered")
hc <- hc_visualize_gene_expression(hc, genes = "CD3D")
```

hc_write_session_info *write session info*

Description

Write session info (S4 API)

Usage

```
hc_write_session_info(hc)
```

Arguments

hc	A HCoCenaExperiment.
----	----------------------

Value

Updated HCoCenaExperiment.

Examples

```
hc <- hc_example_data("prepared")
hc <- hc_write_session_info(hc)
```

HCoCenaClasses	<i>hCoCena S4 Classes</i>
----------------	---------------------------

Description

S4 container classes used by the object-oriented hCoCena API.

plot.hc_sample_regrouping	<i>Plot sample-regrouping heatmaps</i>
---------------------------	--

Description

Draws one or all heatmaps stored in the result returned by [hc_regroup_samples\(\)](#). This uses explicit grid drawing so it also works in RStudio and notebook contexts where `print()` may not display a silent pheatmap object.

Usage

```
## S3 method for class 'hc_sample_regrouping'
plot(x, layer = "all", ...)
```

Arguments

<code>x</code>	An <code>hc_sample_regrouping</code> result.
<code>layer</code>	Layer to draw. Use "all", a result layer id, or a numeric result-layer index.
<code>...</code>	Reserved for future plotting options.

Value

Invisibly returns `x`.

Examples

```
hc <- hc_example_data("clustered")
rg <- hc_regroup_samples(hc, layer = 1, k = 2, apply = FALSE)
plot(rg)
```

Index

* classes

HCoCenaClasses, [137](#)

base::grepl(), [42](#)
base::log(), [7](#)

clusterProfiler::enricher(), [16](#)

hc_add_meta_time_grouping, [5](#)
hc_algo_alluvial, [6](#)
hc_antilog, [7](#)
hc_auto_set_paths, [7](#)
hc_auto_tune, [9](#)
hc_auto_tune_cutoff_tiered, [11](#)
hc_build_integrated_network, [12](#)
hc_build_integrated_network(), [26](#)
hc_celltype_activity_decoupler, [13](#)
hc_celltype_annotation, [15](#)
hc_change_grouping_parameter, [17](#)
hc_check_dirs, [18](#)
hc_check_tf, [19](#)
hc_cluster_calculation, [20](#)
hc_cluster_calculation(), [26](#)
hc_clusters, [21](#)
hc_clusters, HCoCenaExperiment-method
(hc_clusters), [21](#)
hc_col_anno_categorical, [22](#)
hc_col_anno_numerical, [22](#)
hc_colour_single_cluster, [23](#)
hc_config, [24](#)
hc_config, HCoCenaExperiment-method
(hc_config), [24](#)
hc_cut_hclust, [24](#)
hc_define_layers, [25](#)
hc_define_layers(), [115](#)
hc_example_data, [26](#)
hc_export_clusters, [27](#)
hc_export_clusters(), [37](#)
hc_export_to_cytoscape, [27](#)
hc_export_to_local_folder, [28](#)
hc_find_hubs, [29](#)
hc_functional_enrichment, [30](#)
hc_functional_enrichment(), [45](#), [92](#)
hc_gene_to_cluster, [34](#)
hc_get_meta_time_col_order, [34](#)
hc_get_module_scores, [35](#)
hc_graph, [36](#)
hc_graph, HCoCenaExperiment-method
(hc_graph), [36](#)
hc_highlight_geneset, [36](#)
hc_import_clusters, [37](#)
hc_import_layout_from_cytoscape, [38](#)
hc_import_layout_from_local_folder, [39](#)
hc_init, [40](#)
hc_init_save_folder, [40](#)
hc_integration, [41](#)
hc_integration, HCoCenaExperiment-method
(hc_integration), [41](#)
hc_layer_results, [41](#)
hc_layer_results, HCoCenaExperiment-method
(hc_layer_results), [41](#)
hc_list_celltype_databases, [42](#)
hc_list_llm_models, [43](#)
hc_list_llm_models(), [46](#)
hc_llm_enrichment, [44](#)
hc_llm_enrichment(), [43](#), [47](#), [83](#), [84](#)
hc_llm_import, [47](#)
hc_llm_import(), [45](#), [46](#), [83](#), [84](#)
hc_longitudinal_endotype_clustering,
[48](#)
hc_longitudinal_endotype_clustering(),
[100](#)
hc_longitudinal_meta_clustering, [49](#)
hc_longitudinal_module_cap, [51](#)
hc_longitudinal_module_means, [53](#)
hc_longitudinal_module_means(), [48](#), [102](#)
hc_longitudinal_step1_module_donor, [54](#)
hc_longitudinal_step1_module_donor_direct,
[56](#)

hc_longitudinal_step2_meta_clustering, 58
 hc_longitudinal_step2_meta_clustering(), 54
 hc_longitudinal_step3_meta_module_trajectories, 59
 hc_longitudinal_step3_meta_module_trajectories(), 54
 hc_longitudinal_workflow_direct, 61
 hc_mae, 63
 hc_mae, HCoCenaExperiment-method (hc_mae), 63
 hc_merge_clusters, 63
 hc_meta_correlation_cat, 64
 hc_meta_correlation_num, 65
 hc_meta_plot, 66
 hc_module_condition_significance, 67
 hc_module_gfc_tables, 68
 hc_network_comparison_1, 70
 hc_network_comparison_2, 71
 hc_pca, 72
 hc_pca_algo_compare, 73
 hc_plot_celltype_annotation_matrix, 73
 hc_plot_cluster_heatmap, 75
 hc_plot_cluster_heatmap(), 17, 44, 70, 76
 hc_plot_cluster_heatmap_view, 75
 hc_plot_cutoffs, 77
 hc_plot_deg_dist, 78
 hc_plot_enrichment_panels, 78
 hc_plot_enrichment_upstream_network, 79
 hc_plot_gfc_network, 82
 hc_plot_integrated_network, 82
 hc_plot_llm_enrichment, 83
 hc_plot_llm_enrichment(), 45, 46
 hc_plot_longitudinal_cap, 86
 hc_plot_longitudinal_endotype_meta_flow, 87
 hc_plot_longitudinal_endotypes, 88
 hc_plot_longitudinal_enrichment_meta_waves, 89
 hc_plot_longitudinal_enrichment_meta_waves(), 54
 hc_plot_longitudinal_enrichment_waves, 91
 hc_plot_longitudinal_enrichment_waves(), 90
 hc_plot_longitudinal_k_criterion, 94
 hc_plot_longitudinal_meta_consensus, 95
 hc_plot_longitudinal_meta_embeddings, 97
 hc_plot_longitudinal_meta_module_waves, 98
 hc_plot_longitudinal_module_clusters, 100
 hc_plot_longitudinal_module_means, 102
 hc_plot_sample_distributions, 103
 hc_preview_celltype_database, 104
 hc_print_longitudinal_endotypes, 105
 hc_read_data, 106
 hc_read_supplementary, 107
 hc_regroup_samples, 108
 hc_regroup_samples(), 137
 hc_resolve_paths, 110
 hc_run_expression_analysis_1, 112
 hc_run_expression_analysis_1(), 26
 hc_run_expression_analysis_2, 113
 hc_run_expression_analysis_2(), 25, 26, 69
 hc_satellite, 114
 hc_satellite, HCoCenaExperiment-method (hc_satellite), 114
 hc_set_cutoff, 115
 hc_set_cutoff(), 26
 hc_set_global_settings, 116
 hc_set_layer_settings, 118
 hc_set_paths, 120
 hc_set_supp_files, 121
 hc_split_modules, 122
 hc_suggest_topvar, 124
 hc_tf_overrep_module, 124
 hc_tf_overrep_network, 125
 hc_tune_cutoff, 126
 hc_tune_resolution, 128
 hc_unsplit_modules, 129
 hc_update_clustering_algorithm, 130
 hc_upstream_inference, 130
 hc_upstream_inference(), 45
 hc_user_specific_cluster_profiling, 134
 hc_visualize_gene_expression, 135
 hc_write_session_info, 136
 HCoCenaClasses, 137
 HCoCenaConfig-class (HCoCenaClasses), 137

HCoCenaExperiment-class
 (HCoCenaClasses), [137](#)
HCoCenaIntegration-class
 (HCoCenaClasses), [137](#)
HCoCenaLayerResult-class
 (HCoCenaClasses), [137](#)
HCoCenaReferences-class
 (HCoCenaClasses), [137](#)

plot.hc_sample_regrouping, [137](#)

stats::dist(), [109](#)
stats::hclust(), [25](#), [109](#)
stats::p.adjust(), [64](#), [65](#), [131](#)

tempdir(), [26](#)