Package: exploreDE (via r-universe)

November 14, 2026

Type Package

Title Interactive Exploration of Differential Expression Results

Version 0.99.0

Description A Shiny application to help users explore and enjoy their 'omics data, and make publication-quality figures. It shows differential expression results from RNA-seq, proteomics and CRISPR screens as ordination plots, heatmaps, volcano plots, box plots, GO and KEGG pathway views, and ORA and GSEA tables. The app reads AnnData ('.h5ad') files in the exploreDE/1.0 layout, which build_explore_h5ad() writes from counts, sample tables, DE tables and enrichment results, and the SummarizedExperiment results of the FGCZ SUSHI and prolfqua pipelines. App settings are saved and restored as JSON session files.

License MIT + file LICENSE

URL https://fgcz.github.io/exploreDE/, https://github.com/fgcz/exploreDE

BugReports https://github.com/fgcz/exploreDE/issues

biocViews Software, ShinyApps, GUI, Visualization, DifferentialExpression, GeneExpression, RNASeq, Transcriptomics, Proteomics, CRISPR, GeneSetEnrichment, Pathways, GO, KEGG

Encoding UTF-8

Depends R (>= 4.5.0)

Imports shiny, shinydashboard, ggpubr, plotly, DESeq2, RColorBrewer, ComplexHeatmap, clusterProfiler, DT, colourpicker, circlize, kableExtra, ggrepel, sortable, waiter, ggprism, ggbeeswarm, rstatix, parallel, shinyjs, shinycssloaders, GGally, Matrix, fresh, scuttle, qs2, vegan, limma, officer, R6, anndataR (>= 1.2.3), rhdf5, jsonlite, curl, SummarizedExperiment, S4Vectors, dplyr, ggplot2, gtools, matrixStats, methods, openxlsx, readr, tibble, xml2, dendextend, rlang, colorspace, cowplot,

enrichplot, ggvenn, scales, tidyr, stats, utils, grDevices, graphics

Suggests BiocManager, png, ggtext, gridtext, glmpca, pathview, KEGGREST, DOSE, testthat (>= 3.0.0), shinytest2, callr, jsonvalidate, systemfonts, ragg, nanoparquet, arrow, knitr, rmarkdown, BiocStyle, withr, devtools

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libglpk-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev perl python3 libx11-dev zlib1g-dev

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

Date/Publication 2026-11-14 18:03:32 UTC

RemoteUrl https://github.com/BiocStaging/exploreDE

RemoteRef HEAD

RemoteSha 0a0839713a11b83aecac2e7b0cc3bbc4543f1398

Contents

build_explore_h5ad . . . 2
ExploreDataset . . . 5
ExploreDatasetH5AD . . . 13
ExploreDatasetSE . . . 16
ExploreDatasetSE_CountQC . . . 19
ExploreDatasetSE_CrisprCountQC . . . 20
ExploreDatasetSE_Proteomics . . . 21
ExploreDatasetSE_RNASeq . . . 22
exploreDE . . . 23
new_explore_dataset . . . 24
validate_explore_dataset . . . 25

Index **26**

`build_explore_h5ad` *Build a canonical exploreDE* `.h5ad` *from user-supplied data*

Description

The source-agnostic "bring your own data" constructor. `x` is a `DESeqDataSet` (its counts, `colData`, `rowData` and factors are used), a `SummarizedExperiment` (every assay under its name, its `colData` and its `rowData`), or the counts themselves (a matrix, or a named list of matrices such as `list(counts=, normalised=, vst=, tpm=)`). Assay and matrix names become layer names: `rawData` and `raw` are raw intensities in proteomics, `transformedData` the analysis-ready ones. DE tables and pathway statistics are supplied as arguments. Only counts + `coldata` + `omics` are required; `coldata` and `rowdata`, when given, replace those of `x`.

Usage

```
build_explore_h5ad(
  x,
  out,
  omics,
  feature_level = NULL,
  coldata = NULL,
  rowdata = NULL,
  de = list(),
  de_columns = NULL,
  rowdata_columns = NULL,
  pathways = list(),
  enrichment_padj_max = 0.1,
  factors = NULL,
  organism = NA_character_,
  ref_build = NA_character_
)
```

Arguments

`x`	A `DESeqDataSet`, a `SummarizedExperiment` with named assays, a counts matrix, or a named list of count matrices.
`out`	Output `.h5ad` path.
`omics`	Required: `"rnaseq"`, `"proteomics"`, or `"crispr"`.
`feature_level`	Optional override of the feature granularity within the omics — rnaseq `"gene"`/`"isoform"`, proteomics `"protein"`/`"peptide"`/`"phospho"`. Omit to auto-detect from `rowdata` (`site` -> phospho, `peptide_Id` -> peptide, `transcript_id` -> isoform).
`coldata`	Sample metadata `data.frame`. Required unless `x` is a rich object that carries it.
`rowdata`	Optional per-feature annotation (feature names/symbols/ids and per-feature pathway annotation columns, e.g. `GO BP`).
`de`	Optional named list of DE tables keyed by contrast (`DESeqResults` / `data.frame`). Name each contrast `<test>_vs_<reference>`, with both groups levels of one `coldata` factor (an error otherwise). Columns are recognised by the column dictionary (`inst/dictionaries/column_patterns.json`), overridable with `de_columns`. An optional extra score column (e.g. SAINTexpress's `SaintScore`) is picked up by the dictionary alone and shown for display under its own column name.

de_columns
: Optional DE column overrides: `list(effect_log2fc = "<col>", score_pvalue = "<col>", score_fdr = "<col>")`, applied to all contrasts, or a named list of such lists keyed by contrast. Named columns are used as given; any not named are still found by the dictionary. A producer with no raw p-value (e.g. SAINT-express) simply has none; the app degrades to FDR-only. Unknown keys are ignored with a warning. Overrides are persisted at `uns$exploreDE$mapping$de`, so the file stays self-describing.

rowdata_columns
: Optional feature column overrides: `list(id_col = "<col>", symbol_col = "<col>")` — which `rowdata` column supplies the feature id and the display symbol. Anything not named is still found by the dictionary. Unknown keys are ignored with a warning. Overrides are persisted at `uns$exploreDE$mapping$feature`.

pathways
: Optional pathway statistics: a named list keyed by contrast (keys a subset of `de`); each entry a named list of clusterProfiler objects and/or `data.frames`.

enrichment_padj_max
: p.adjust cut-off applied to clusterProfiler results before they are stored (`p.adjust <= enrichment_padj_max`). Either one number for every method, or a named list keyed by the method names used in `pathways` (e.g. `list(ora = 0.05, gsea = 0.05)`) so that ORA and GSEA can follow their own FDR thresholds; a method not named in the list uses the largest value given. The value(s) are recorded in `uns$enrichment_filter` and the app uses a per-method value as its significance threshold.

factors
: Optional colData columns that are experimental factors (default: all character/factor-typed columns).

organism, ref_build
: Optional labels.

Value

The output path, invisibly.

Examples

```
library(SummarizedExperiment)
# A prolfqua proteomics result: one contrast, nested in rowData
se <- readRDS(system.file("examples/proteomics/SummarizedExperiment.rds", package = "exploreDE"))
de <- as.data.frame(rowData(se)$constrast_B_vs_A)
path <- build_explore_h5ad(
  x = list(raw = assay(se, "rawData"), transformed = assay(se, "transformedData")),
  out = tempfile(fileext = ".h5ad"),
  omics = "proteomics",
  coldata = colData(se),
  rowdata = de,
  factors = "G_",
  de = list(B_vs_A = de)
)
file.exists(path)

# The SummarizedExperiment itself: rawData and transformedData become the
```

```
# layers intensities_raw and intensities
se_path <- build_explore_h5ad(
  se,
  out = tempfile(fileext = ".h5ad"),
  omics = "proteomics",
  rowdata = de,
  factors = "G_",
  de = list(B_vs_A = de)
)

# Counts only
counts_only <- build_explore_h5ad(
  assay(se, "rawData"),
  out = tempfile(fileext = ".h5ad"),
  omics = "proteomics",
  coldata = colData(se)
)
```

`ExploreDataset` *Abstract base class for exploreDE's storage-agnostic dataset wrapper*

Description

`ExploreDataset` defines the accessor contract that every concrete subclass must implement. It is not instantiable on its own — calling `ExploreDataset$new()` raises an error. Use `new_explore_dataset()` to build a concrete instance, or instantiate a subclass directly.

Methods implemented on the base class read only from `private$.config` and are inherited unchanged by every subclass. Methods that touch the storage backend are virtual (`stop("not implemented")`) and must be overridden.

Value

An R6 class generator. `$new()` always errors; the concrete subclasses return an object inheriting from `ExploreDataset`.

Methods

Public methods:

- `ExploreDataset$new()`
- `ExploreDataset$omics_class()`
- `ExploreDataset$data_type()`
- `ExploreDataset$has_de()`
- `ExploreDataset$has_enrichment()`
- `ExploreDataset$organism()`
- `ExploreDataset$ref_build()`
- `ExploreDataset$feature_id_kind()`
- `ExploreDataset$background_expression()`

- ExploreDataset$backend_kind()
- ExploreDataset$backend()
- ExploreDataset$samples()
- ExploreDataset$sample_ids()
- ExploreDataset$n_samples()
- ExploreDataset$factor_names()
- ExploreDataset$factor_levels()
- ExploreDataset$primary_grouping()
- ExploreDataset$secondary_grouping()
- ExploreDataset$sample_factor()
- ExploreDataset$features()
- ExploreDataset$feature_ids()
- ExploreDataset$n_features()
- ExploreDataset$feature_display()
- ExploreDataset$assay_names()
- ExploreDataset$has_assay()
- ExploreDataset$assay()
- ExploreDataset$assay_label()
- ExploreDataset$assay_is_log2()
- ExploreDataset$raw_assay_name()
- ExploreDataset$primary_assay_name()
- ExploreDataset$raw_assay()
- ExploreDataset$primary_assay()
- ExploreDataset$contrasts()
- ExploreDataset$primary_contrast()
- ExploreDataset$de()
- ExploreDataset$de_all()
- ExploreDataset$contrast_meta()
- ExploreDataset$contrast_label()
- ExploreDataset$de_display_extras()
- ExploreDataset$has_pvalue()
- ExploreDataset$enrichment_kinds()
- ExploreDataset$enrichment()
- ExploreDataset$enrichment_input()
- ExploreDataset$enrichment_table()
- ExploreDataset$enrichment_filter()
- ExploreDataset$sgrna_to_gene()
- ExploreDataset$proteomics_choices()
- ExploreDataset$all_pathways()
- ExploreDataset$source()
- ExploreDataset$format()
- ExploreDataset$clone()

`ExploreDataset$new()`: Always errors: the base class is abstract. Use `new_explore_dataset()` or a concrete subclass.

Usage:

```
ExploreDataset$new(...)
```

Arguments:

`...` Ignored.

`ExploreDataset$omics_class()`: Omics class from the config: `"genomics"` or `"proteomics"`.

Usage:

```
ExploreDataset$omics_class()
```

`ExploreDataset$data_type()`: Dataset type from the config, e.g. `"RNASeq"`, `"CountQC"`, `"CrisprCountQC"` or `"proteomics"`.

Usage:

```
ExploreDataset$data_type()
```

`ExploreDataset$has_de()`: `TRUE` when the dataset has at least one contrast.

Usage:

```
ExploreDataset$has_de()
```

`ExploreDataset$has_enrichment()`: `TRUE` when the config lists at least one enrichment kind.

Usage:

```
ExploreDataset$has_enrichment()
```

`ExploreDataset$organism()`: Organism from the config, or `NA` when unknown.

Usage:

```
ExploreDataset$organism()
```

`ExploreDataset$ref_build()`: Reference genome build from the config, or `NA` when unknown.

Usage:

```
ExploreDataset$ref_build()
```

`ExploreDataset$feature_id_kind()`: What a feature ID denotes, from the config, e.g. `"gene"`, `"isoform"`, `"protein"`, `"peptide"`, `"phospho"` or `"guide"`.

Usage:

```
ExploreDataset$feature_id_kind()
```

`ExploreDataset$background_expression()`: Low-signal offset added to expression values before taking log2; from the config, default 10.

Usage:

```
ExploreDataset$background_expression()
```

`ExploreDataset$backend_kind()`: Short tag naming the storage backend (`"se"`, `"h5ad"` or `"parquet"`). Implemented by subclasses.

Usage:

```
ExploreDataset$backend_kind()
```

`ExploreDataset$backend()`: The underlying storage object. Implemented by subclasses.

Usage:

```
ExploreDataset$backend()
```

`ExploreDataset$samples()`: Sample metadata as a data.frame, one row per sample, rownames are sample IDs. Implemented by subclasses.

Usage:

```
ExploreDataset$samples()
```

`ExploreDataset$sample_ids()`: Sample IDs: the rownames of `samples()`.

Usage:

```
ExploreDataset$sample_ids()
```

`ExploreDataset$n_samples()`: Number of samples: the number of rows of `samples()`.

Usage:

```
ExploreDataset$n_samples()
```

`ExploreDataset$factor_names()`: Names of the experimental factors from the config; empty when there are none.

Usage:

```
ExploreDataset$factor_names()
```

`ExploreDataset$factor_levels()`: Levels of one experimental factor, from the config.

Usage:

```
ExploreDataset$factor_levels(factor)
```

Arguments:

`factor` Factor name, one of `factor_names()`.

`ExploreDataset$primary_grouping()`: Name of the factor used as the primary sample grouping, from the config.

Usage:

```
ExploreDataset$primary_grouping()
```

`ExploreDataset$secondary_grouping()`: Name of the secondary grouping factor, or `NA` when there is none.

Usage:

```
ExploreDataset$secondary_grouping()
```

`ExploreDataset$sample_factor()`: One column of `samples()`: the values of a factor, one per sample.

Usage:

```
ExploreDataset$sample_factor(factor)
```

Arguments:

factor Column name in samples(), usually one of factor_names().

ExploreDataset$features(): Feature annotation as a data.frame, one row per feature, rownames are feature IDs, with a display_name column. Implemented by subclasses.

Usage:

```
ExploreDataset$features()
```

ExploreDataset$feature_ids(): Feature IDs: the rownames of features().

Usage:

```
ExploreDataset$feature_ids()
```

ExploreDataset$n_features(): Number of features: the number of rows of features().

Usage:

```
ExploreDataset$n_features()
```

ExploreDataset$feature_display(): Display names of features, named by feature ID.

Usage:

```
ExploreDataset$feature_display(ids = NULL)
```

Arguments:

ids Feature IDs to look up; NULL (default) returns all features.

ExploreDataset$assay_names(): Canonical names of the assays this dataset provides. Implemented by subclasses.

Usage:

```
ExploreDataset$assay_names()
```

ExploreDataset$has_assay(): TRUE when name is one of assay_names().

Usage:

```
ExploreDataset$has_assay(name)
```

Arguments:

name Canonical assay name.

ExploreDataset$assay(): One assay as a features x samples matrix, optionally subset. Implemented by subclasses.

Usage:

```
ExploreDataset$assay(name, features = NULL, samples = NULL)
```

Arguments:

name Canonical assay name, one of assay_names().

features Feature IDs (rows) to keep; NULL keeps all.

samples Sample IDs (columns) to keep; NULL keeps all.

ExploreDataset$assay_label(): Human-readable label for an assay, from the config; name itself when no label is registered.

Usage:

`ExploreDataset$assay_label(name)`

Arguments:

`name` Canonical assay name.

`ExploreDataset$assay_is_log2()`: `TRUE` when an assay is on the log2 scale (log2 counts, VST, transformed intensities), from the config. Errors on an assay the dataset does not have.

Usage:

`ExploreDataset$assay_is_log2(name)`

Arguments:

`name` Canonical assay name, one of `assay_names()`.

`ExploreDataset$raw_assay_name()`: Canonical name of the raw assay, from the config.

Usage:

`ExploreDataset$raw_assay_name()`

`ExploreDataset$primary_assay_name()`: Canonical name of the primary assay, from the config.

Usage:

`ExploreDataset$primary_assay_name()`

`ExploreDataset$raw_assay()`: The raw assay: shortcut for `assay(raw_assay_name(), ...)`.

Usage:

`ExploreDataset$raw_assay(...)`

Arguments:

`...` Passed to `assay()`: `features`, `samples`.

`ExploreDataset$primary_assay()`: The primary assay: shortcut for `assay(primary_assay_name(), ...)`.

Usage:

`ExploreDataset$primary_assay(...)`

Arguments:

`...` Passed to `assay()`: `features`, `samples`.

`ExploreDataset$contrasts()`: Names of the DE contrasts, from the config; empty when the dataset has no DE results.

Usage:

`ExploreDataset$contrasts()`

`ExploreDataset$primary_contrast()`: Name of the primary contrast, from the config; `NA` when there are no contrasts.

Usage:

`ExploreDataset$primary_contrast()`

`ExploreDataset$de()`: DE result table for one contrast, as a data.frame. Implemented by subclasses.

Usage:

```
ExploreDataset$de(contrast)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`ExploreDataset$de_all()`: DE result tables of all contrasts, as a list named by contrast.

Usage:

```
ExploreDataset$de_all()
```

`ExploreDataset$contrast_meta()`: Metadata list of one contrast from the config: test and reference group, factor, method, thresholds. Errors on an unknown contrast.

Usage:

```
ExploreDataset$contrast_meta(contrast)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`ExploreDataset$contrast_label()`: Display label of a contrast: currently the contrast name itself. Errors on an unknown contrast.

Usage:

```
ExploreDataset$contrast_label(contrast)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`ExploreDataset$de_display_extras()`: Names of extra DE columns shown for display only (e.g. `SaintScore`), from the config; empty by default.

Usage:

```
ExploreDataset$de_display_extras()
```

`ExploreDataset$has_pvalue()`: `TRUE` when the DE table of at least one contrast has a non-missing raw `pValue`; `FALSE` for FDR-only producers such as SAINTexpress.

Usage:

```
ExploreDataset$has_pvalue()
```

`ExploreDataset$enrichment_kinds()`: Enrichment kinds available, from the config: any of `"ora"` and `"gsea"`.

Usage:

```
ExploreDataset$enrichment_kinds()
```

`ExploreDataset$enrichment()`: Enrichment results of one contrast and kind, as a flat named list of clusterProfiler results. Implemented by subclasses.

Usage:

```
ExploreDataset$enrichment(contrast, kind)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`kind` `"ora"` or `"gsea"`.

`ExploreDataset$enrichment_input()`: Inputs for enrichment of one contrast: `list(seq_anno, ranked_l2fc, selections)`. Implemented by subclasses.

Usage:

```
ExploreDataset$enrichment_input(contrast)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`ExploreDataset$enrichment_table()`: Enrichment results of one contrast and kind as one tidy data.frame, derived from `enrichment()`; an empty data.frame when there are none.

Usage:

```
ExploreDataset$enrichment_table(contrast, kind)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`kind` `"ora"` or `"gsea"`.

`ExploreDataset$enrichment_filter()`: The `p.adjust` cut-off applied to the stored enrichment tables, from the config; `NA` when none was applied.

Usage:

```
ExploreDataset$enrichment_filter()
```

`ExploreDataset$sgrna_to_gene()`: Named character vector mapping sgRNA IDs to gene names (CRISPR datasets); `NULL` otherwise.

Usage:

```
ExploreDataset$sgrna_to_gene()
```

`ExploreDataset$proteomics_choices()`: Choices and lookup for the proteomics feature picker, `list(choices, lookup)`; `NULL` when the dataset has none.

Usage:

```
ExploreDataset$proteomics_choices(kind = c("gene", "protein"))
```

Arguments:

`kind` `"gene"` (default) or `"protein"`.

`ExploreDataset$all_pathways()`: GO terms named in the `GO BP`, `GO MF` and `GO CC` columns of `features()`, as `"GO:ID term name"` strings; empty when those columns are absent.

Usage:

```
ExploreDataset$all_pathways()
```

`ExploreDataset$source()`: Provenance list from the config: backend kind, schema version and build time.

Usage:

```
ExploreDataset$source()
```

`ExploreDataset$format()`: Summary of the dataset (type, dimensions, factors, contrasts, assays, enrichment), one line per element. `print()` on the object shows these lines.

Usage:

```
ExploreDataset$format(...)
```

Arguments:

`...` Ignored.

`ExploreDataset$clone()`: The objects of this class are cloneable with this method.

Usage:

```
ExploreDataset$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
se <- readRDS(system.file("examples/proteomics/SummarizedExperiment.rds", package = "exploreDE"))
eds <- new_explore_dataset(se)
inherits(eds, "ExploreDataset")
eds$contrasts()
head(eds$de(eds$primary_contrast()))

# The base class itself cannot be instantiated
try(ExploreDataset$new())
```

ExploreDatasetH5AD *Concrete* ExploreDataset *backend over a canonical AnnData (*.h5ad*)*

Description

Reads a canonical exploreDE/1.0 AnnData and implements the accessor contract inherited from ExploreDataset. Accepts a path to a `.h5ad` file or a live `anndataR` object. This is the backend for user-supplied data ("bring your own data") and the eventual single backend once producers emit canonical `.h5ad` directly.

Value

An R6 class generator; `$new()` returns an `ExploreDatasetH5AD` object.

Super class

`ExploreDataset` -> `ExploreDatasetH5AD`

Methods

Public methods:

- ExploreDatasetH5AD$new()
- ExploreDatasetH5AD$backend()
- ExploreDatasetH5AD$backend_kind()
- ExploreDatasetH5AD$samples()
- ExploreDatasetH5AD$features()
- ExploreDatasetH5AD$de()
- ExploreDatasetH5AD$assay_names()
- ExploreDatasetH5AD$assay()
- ExploreDatasetH5AD$enrichment()
- ExploreDatasetH5AD$enrichment_table()
- ExploreDatasetH5AD$enrichment_input()
- ExploreDatasetH5AD$clone()

`ExploreDatasetH5AD$new()`: Read a canonical `.h5ad` or take a live AnnData, build the config from it, and prepare the feature and DE tables.

Usage:

```
ExploreDatasetH5AD$new(input)
```

Arguments:

`input` An `anndataR` AnnData object, or a path to an existing `.h5ad` file.

`ExploreDatasetH5AD$backend()`: The AnnData parts read at construction, as `list(obs, var, layers, varm, uns)` of plain R objects.

Usage:

```
ExploreDatasetH5AD$backend()
```

`ExploreDatasetH5AD$backend_kind()`: Returns `"h5ad"`.

Usage:

```
ExploreDatasetH5AD$backend_kind()
```

`ExploreDatasetH5AD$samples()`: Sample metadata from `obs`, with the `factor_` prefix stripped from column names and a `Name` column added when missing.

Usage:

```
ExploreDatasetH5AD$samples()
```

`ExploreDatasetH5AD$features()`: Feature annotation from `var`, with `feature_id` and `display_name` columns, ordered by the p-value of the primary contrast.

Usage:

```
ExploreDatasetH5AD$features()
```

`ExploreDatasetH5AD$de()`: DE table of one contrast from `varm`, with the legacy aliases `log2Ratio`, `pValue` and `fdr` added; an empty data.frame when the slot is missing. Errors on an unknown contrast.

Usage:

```
ExploreDatasetH5AD$de(contrast)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`ExploreDatasetH5AD$assay_names()`: Canonical assay names from the config's assay map whose layer exists; computed assays are always included.

Usage:

```
ExploreDatasetH5AD$assay_names()
```

`ExploreDatasetH5AD$assay()`: One assay as a features x samples matrix (the layer transposed); `normalised`, `normalised_log2`, `vst`, `fpkm` and `tpm` are computed when the config maps them to no layer.

Usage:

```
ExploreDatasetH5AD$assay(name, features = NULL, samples = NULL)
```

Arguments:

`name` Canonical assay name, one of `assay_names()`.

`features` Feature IDs (rows) to keep; `NULL` keeps all.

`samples` Sample IDs (columns) to keep; `NULL` keeps all.

`ExploreDatasetH5AD$enrichment()`: Enrichment results of one contrast and kind, rebuilt from `uns$enrichment` and `uns$enrichment_support`.

Usage:

```
ExploreDatasetH5AD$enrichment(contrast, kind)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`kind` `"ora"` or `"gsea"`.

`ExploreDatasetH5AD$enrichment_table()`: Tidy enrichment table of one contrast and kind: the table stored in `uns$enrichment` when present, otherwise derived from `enrichment()` as in the base class.

Usage:

```
ExploreDatasetH5AD$enrichment_table(contrast, kind)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`kind` `"ora"` or `"gsea"`.

`ExploreDatasetH5AD$enrichment_input()`: Enrichment inputs rebuilt from the DE table: `features()`, log2 fold changes sorted decreasing, and up/down feature selections at the contrast's FDR and log2FC thresholds.

Usage:

```
ExploreDatasetH5AD$enrichment_input(contrast)
```

Arguments:

contrast Contrast name, one of contrasts().

ExploreDatasetH5AD$clone(): The objects of this class are cloneable with this method.

Usage:

```
ExploreDatasetH5AD$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
library(SummarizedExperiment)
se <- readRDS(system.file("examples/proteomics/SummarizedExperiment.rds", package = "exploreDE"))
de <- as.data.frame(rowData(se)$constrast_B_vs_A)
path <- tempfile(fileext = ".h5ad")
build_explore_h5ad(
  x = list(raw = assay(se, "rawData"), transformed = assay(se, "transformedData")),
  out = path,
  omics = "proteomics",
  coldata = colData(se),
  rowdata = de,
  factors = "G_",
  de = list(B_vs_A = de)
)
eds <- ExploreDatasetH5AD$new(path)
eds
head(eds$de("B_vs_A"))
```

ExploreDatasetSE *Abstract intermediate class for SE-backed datasets*

Description

ExploreDatasetSE holds the SummarizedExperiment backend reference and implements the accessor contract uniformly for every SE shape. Concrete subclasses (ExploreDatasetSE_RNASeq, _CountQC, _CrisprCountQC, _Proteomics) override two private methods only:

- .build_config(se) — returns the per-instance config list at $initialize() time
- .prep() — populates private$.features_df, private$.de_dfs, private$.fid_lookup from the SE

This class is not instantiable on its own — instantiate a concrete subclass instead.

Value

An R6 class generator; the concrete subclasses' $new() returns an object inheriting from ExploreDatasetSE.

Super class

ExploreDataset -> ExploreDatasetSE

Methods

Public methods:

- ExploreDatasetSE$new()
- ExploreDatasetSE$backend()
- ExploreDatasetSE$backend_kind()
- ExploreDatasetSE$samples()
- ExploreDatasetSE$features()
- ExploreDatasetSE$de()
- ExploreDatasetSE$assay_names()
- ExploreDatasetSE$assay()
- ExploreDatasetSE$enrichment()
- ExploreDatasetSE$enrichment_input()
- ExploreDatasetSE$clone()

`ExploreDatasetSE$new()`: Wrap a `SummarizedExperiment`: store it, build the config from it, and prepare the feature and DE tables.

Usage:

```
ExploreDatasetSE$new(se)
```

Arguments:

`se` A `SummarizedExperiment`.

`ExploreDatasetSE$backend()`: The wrapped `SummarizedExperiment`.

Usage:

```
ExploreDatasetSE$backend()
```

`ExploreDatasetSE$backend_kind()`: Returns `"se"`.

Usage:

```
ExploreDatasetSE$backend_kind()
```

`ExploreDatasetSE$samples()`: Sample metadata from `colData()`, with the SUSHI `[Factor]` suffix stripped from column names, rownames set to sample IDs, and a `Name` column added.

Usage:

```
ExploreDatasetSE$samples()
```

`ExploreDatasetSE$features()`: Feature annotation table, prepared from `rowData()` by the subclass.

Usage:

```
ExploreDatasetSE$features()
```

`ExploreDatasetSE$de()`: DE table of one contrast, prepared from `rowData()` by the subclass; an empty data.frame when none was prepared. Errors on an unknown contrast.

Usage:

```
ExploreDatasetSE$de(contrast)
```

Arguments:

`contrast` Contrast name, one of `contrasts()`.

`ExploreDatasetSE$assay_names()`: Canonical assay names from the config's assay map whose SE assay exists; computed assays are always included.

Usage:

```
ExploreDatasetSE$assay_names()
```

`ExploreDatasetSE$assay()`: One assay as a features x samples matrix with rows renamed to feature IDs; `normalised_log2`, `vst`, `fpkm` and `tpm` are computed when the config maps them to no SE assay.

Usage:

```
ExploreDatasetSE$assay(name, features = NULL, samples = NULL)
```

Arguments:

`name` Canonical assay name, one of `assay_names()`.

`features` Feature IDs (rows) to keep; `NULL` keeps all.

`samples` Sample IDs (columns) to keep; `NULL` keeps all.

`ExploreDatasetSE$enrichment()`: Enrichment results of one kind, read from `metadata(se)` and flattened to a named list; an empty list when absent. `contrast` is not used.

Usage:

```
ExploreDatasetSE$enrichment(contrast, kind)
```

Arguments:

`contrast` Contrast name; not used by this backend.

`kind` `"ora"` or `"gsea"`.

`ExploreDatasetSE$enrichment_input()`: Enrichment inputs read from `metadata(se)$enrichInput`, with `NULL` fields and no selections when absent. `contrast` is not used.

Usage:

```
ExploreDatasetSE$enrichment_input(contrast)
```

Arguments:

`contrast` Contrast name; not used by this backend.

`ExploreDatasetSE$clone()`: The objects of this class are cloneable with this method.

Usage:

```
ExploreDatasetSE$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
se <- readRDS(system.file("examples/proteomics/SummarizedExperiment.rds", package = "exploreDE"))
eds <- new_explore_dataset(se)
inherits(eds, "ExploreDatasetSE")
eds$assay_names()
dim(eds$raw_assay())
```

ExploreDatasetSE_CountQC *Concrete subclass for SUSHI CountQC* SummarizedExperiment *outputs*

Description

Top-level rowData(se) carries feature annotations; there are no DE results. May include a presentFlag / isPresent assay used as the QC denominator.

Details

Inherits the accessor contract from ExploreDataset via ExploreDatasetSE; overrides only .build_config() and .prep().

Value

An R6 class generator; $new() returns an ExploreDatasetSE_CountQC object.

Super classes

ExploreDataset -> ExploreDatasetSE -> ExploreDatasetSE_CountQC

Methods

Public methods:

- ExploreDatasetSE_CountQC$clone()

ExploreDatasetSE_CountQC$clone(): The objects of this class are cloneable with this method.

Usage:

```
ExploreDatasetSE_CountQC$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
library(SummarizedExperiment)
de_result <- readRDS(system.file("examples/genomics/deResult.rds", package = "exploreDE"))
# A count table without DE results, as SUSHI's CountQC writes it
se <- SummarizedExperiment(
  assays = list(counts = assay(de_result, "counts")),
  colData = colData(de_result),
  rowData = rowData(de_result)[, c("gene_id", "gene_name", "description")]
)
eds <- ExploreDatasetSE_CountQC$new(se)
eds
eds$has_de()
```

ExploreDatasetSE_CrisprCountQC *Concrete subclass for CRISPR Mageck CountQC outputs*

Description

Identical to ExploreDatasetSE_CountQC plus a sgRNA → gene mapping parsed from the SE feature IDs, which are assumed to follow the GENE_<n>_<SEQUENCE> convention.

Details

Inherits .prep() from ExploreDatasetSE_CountQC unchanged.

Value

An R6 class generator; $new() returns an ExploreDatasetSE_CrisprCountQC object.

Super classes

ExploreDataset -> ExploreDatasetSE -> ExploreDatasetSE_CountQC -> ExploreDatasetSE_CrisprCountQC

Methods

Public methods:

- ExploreDatasetSE_CrisprCountQC$clone()

ExploreDatasetSE_CrisprCountQC$clone(): The objects of this class are cloneable with this method.

Usage:

```
ExploreDatasetSE_CrisprCountQC$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
library(SummarizedExperiment)
# Guide counts: 4 guides for each of 10 genes, named GENE_<n>_<SEQUENCE>
set.seed(1)
guides <- sprintf("GENE%d_%d_ACGTACGTACGT", rep(1:10, each = 4), rep(1:4, 10))
counts <- matrix(rpois(40 * 4, 50), nrow = 40, dimnames = list(guides, paste0("S", 1:4)))
samples <- S4Vectors::DataFrame(
  `Condition [Factor]` = rep(c("ctrl", "treated"), each = 2),
  row.names = colnames(counts),
  check.names = FALSE
)
se <- SummarizedExperiment(assays = list(counts = counts), colData = samples)
eds <- ExploreDatasetSE_CrisprCountQC$new(se)
eds
head(eds$sgrna_to_gene())
```

ExploreDatasetSE_Proteomics *Concrete subclass for FGCZ proteomics* SummarizedExperiment *outputs*

Description

Reads from a prolfqua-derived SE. Top-level rowData(se) is sparse; feature metadata and per-contrast DE statistics live nested inside rowData(se)$constrast_<name> DataFrames (note the SUSHI typo is preserved). The first contrast is the source of truth for feature metadata. feature_id is **built** from protein_Id (+ site / peptide_Id to disambiguate), and the assay matrices' rownames are remapped via .fid_lookup since SE rownames do not match the built feature_ids.

Details

Inherits the accessor contract from ExploreDataset via ExploreDatasetSE; overrides .build_config() and .prep().

Value

An R6 class generator; $new() returns an ExploreDatasetSE_Proteomics object.

Super classes

ExploreDataset -> ExploreDatasetSE -> ExploreDatasetSE_Proteomics

Methods

Public methods:

- ExploreDatasetSE_Proteomics$clone()

ExploreDatasetSE_Proteomics$clone(): The objects of this class are cloneable with this method.

Usage:

```
ExploreDatasetSE_Proteomics$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
se <- readRDS(system.file("examples/proteomics/SummarizedExperiment.rds", package = "exploreDE"))
eds <- ExploreDatasetSE_Proteomics$new(se)
eds
head(eds$feature_ids())
head(eds$de(eds$primary_contrast()))
```

ExploreDatasetSE_RNASeq *Concrete subclass for SUSHI RNA-seq* SummarizedExperiment *outputs*

Description

Reads from a SUSHI ezApp DESeq2 / edgeR / limma result. Top-level `rowData(se)` carries feature annotations plus inline DE columns for the single contrast described by `param$comparison`. Optional pre-computed `enrichResult` / `gseaResult` objects live in `metadata(se)`.

Details

Inherits the accessor contract from ExploreDataset via ExploreDatasetSE; overrides only `.build_config()` and `.prep()`.

Value

An R6 class generator; `$new()` returns an `ExploreDatasetSE_RNASeq` object.

Super classes

ExploreDataset -> ExploreDatasetSE -> ExploreDatasetSE_RNASeq

Methods

Public methods:

- ExploreDatasetSE_RNASeq$clone()

`ExploreDatasetSE_RNASeq$clone()`: The objects of this class are cloneable with this method.

Usage:

```
ExploreDatasetSE_RNASeq$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
se <- readRDS(system.file("examples/genomics/deResult.rds", package = "exploreDE"))
eds <- ExploreDatasetSE_RNASeq$new(se)
eds
eds$contrasts()
head(eds$de(eds$primary_contrast()))
eds$enrichment_kinds()
```

`exploreDE` *Start the exploreDE app*

Description

Opens the app locally. With no `file`, it starts on the landing page, where you can upload one or more `.h5ad` files built with `build_explore_h5ad()` or load the demo dataset. With `file`, it opens straight into that dataset; with several files, the first opens in the tabs and the contrasts of the others join it in the Multi DE tabs.

Usage

```
exploreDE(file = NULL, ...)
```

Arguments

`file`	Optional dataset: a `SummarizedExperiment` or AnnData object; a path to an exploreDE `.h5ad`, a SummarizedExperiment `.rds`/`.qs2`, or a SUSHI result directory; or several paths, the first of which opens in the tabs, the others going to Multi DE. An object is written to a temporary file, which the app opens.
`...`	Passed to `shiny::runApp()` (e.g. `port`, `launch.browser`).

Details

A `SummarizedExperiment` opens as `new_explore_dataset()` classifies it: a SUSHI RNA-seq or prolfqua proteomics result with all its tabs, any other as a Count QC dataset. To see the DE tabs for another `SummarizedExperiment`, build an `.h5ad` from it with `build_explore_h5ad()` and open that.

Value

Called to run the app; returns the value passed to `shiny::stopApp()` when the app stops (usually `NULL`), as `shiny::runApp()` does.

Examples

```
if (interactive()) {
  exploreDE()
  exploreDE("my_data.h5ad")
  exploreDE(c("design1.h5ad", "design2.h5ad"))
 se <- readRDS(system.file("examples/proteomics/SummarizedExperiment.rds", package = "exploreDE"))
  exploreDE(se)
}
```

`new_explore_dataset` *Construct an* `ExploreDataset` *instance from a path, URL, or SE*

Description

Factory function: loads the input (or accepts a live `SummarizedExperiment`), classifies its shape via `detect_se_shape()`, and returns the appropriate concrete subclass.

Usage

```
new_explore_dataset(input)
```

Arguments

`input`	See above.

Details

Accepted inputs:

- A live `SummarizedExperiment` object.
- A character path to an `.rds` / `.qs2` file.
- A character path to a `.RData` file (Mageck EzResult convention).
- A character path to a directory containing one of the standard SUSHI artefacts (`deResult.rds`, `rawData_final.rds`, etc.).
- An `http(s)://` URL pointing at an `.rds` file.

Value

An object inheriting from ExploreDataset.

Examples

```
# From a live SummarizedExperiment
se <- readRDS(system.file("examples/proteomics/SummarizedExperiment.rds", package = "exploreDE"))
eds <- new_explore_dataset(se)
class(eds)[1]

# From an .h5ad file written by build_explore_h5ad()
path <- build_explore_h5ad(
  SummarizedExperiment::assay(se, "rawData"),
  out = tempfile(fileext = ".h5ad"),
  omics = "proteomics",
  coldata = SummarizedExperiment::colData(se)
)
eds_h5ad <- new_explore_dataset(path)
class(eds_h5ad)[1]
eds_h5ad$has_de()
```

validate_explore_dataset *Runtime validator for a live* ExploreDataset *instance*

Description

Checks subclass-agnostic invariants by walking the accessor API:

Usage

```
validate_explore_dataset(eds, strict = TRUE)
```

Arguments

eds	An ExploreDataset instance.
strict	If TRUE (default), stop() listing every failure. If FALSE, returns list(ok, errors) and never throws: a failing accessor is recorded as an error.

Details

1. The runtime class is a concrete ExploreDataset subclass — not the abstract base or intermediate.
2. Cross-references resolve: factor_names() are columns of samples(); every contrasts() entry has a working de() with the effect_log2fc and score_fdr columns; primary_grouping() is a member of factor_names(); primary_contrast() is in contrasts() (or NA).
3. Assays: raw_assay_name() and primary_assay_name() are in assay_names(), every assay has a log2 fact (assay_is_log2()), and the raw assay's rownames / colnames agree with feature_ids() / sample_ids().

Cost is proportional to the number of contrasts (one de(c) round-trip each) plus a small constant of other accessor calls.

Value

Invisibly, TRUE on success (strict mode); or list(ok = logical(1), errors = character) (non-strict mode).

Examples

```
se <- readRDS(system.file("examples/proteomics/SummarizedExperiment.rds", package = "exploreDE"))
eds <- new_explore_dataset(se)
validate_explore_dataset(eds)
validate_explore_dataset(eds, strict = FALSE)
```

Index

build_explore_h5ad, 2
build_explore_h5ad(), *23*

ExploreDataset, 5, *13*, *16*, *19–22*, *24*
ExploreDatasetH5AD, 13
ExploreDatasetSE, 16, *19–22*
ExploreDatasetSE_CountQC, 19, *20*
ExploreDatasetSE_CrisprCountQC, 20
ExploreDatasetSE_Proteomics, 21
ExploreDatasetSE_RNASeq, 22
exploreDE, 23

new_explore_dataset, 24
new_explore_dataset(), *5*, *7*, *23*

shiny::runApp(), *23*
shiny::stopApp(), *23*

validate_explore_dataset, 25