Package 'scOverlay'

November 6, 2026

Type Package

BiocType Software

Title Multilayer Plots for Single-Cell Data

Version 0.99.5

Description scOverlay provides publication-grade plotting helpers to display a foreground feature (gene expression or metadata) on top of a contextual background (clusters, cell types, samples, or a solid colour) in the same reduced dimension panel, each with independent colour scales (via ggnewscale). It integrates with SingleCellExperiment / SummarizedExperiment, supports flexible subsetting of foreground cells, multiple assays (e.g. counts, logcounts), and grid arrangements by user-defined grouping columns.

License Artistic-2.0

Encoding UTF-8

Depends R (>= 4.6.0)

Imports ggplot2, ggnewscale, SummarizedExperiment, SingleCellExperiment, viridisLite, scales, RColorBrewer, patchwork, ggrastr, rlang

Suggests knitr, rmarkdown, testthat (>= 3.0.0), BiocStyle

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

URL https://github.com/bernatgel/scOverlay

BugReports https://github.com/bernatgel/scOverlay/issues

biocViews Software, Visualization, SingleCell, Transcriptomics, GeneExpression, DataRepresentation

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

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

git_branch devel

git_last_commit 17dd0b4

git_last_commit_date 2026-10-20

Repository Bioconductor 3.24

Date/Publication 2026-11-05

Author Bernat Gel [aut, cre] (ORCID: <https://orcid.org/0000-0001-8878-349X>)

Maintainer Bernat Gel <bgel@igtp.cat>

Contents

arrangeNestedPlots . . . 2
getPalette . . . 3
listPalettes . . . 5
plotBivariateLegend . . . 6
plotGeneOverlay . . . 7
plotOverlay . . . 8
plotOverlayPerGroup . . . 13
plotOverlayPerNestedGroup . . . 18
plotPalettes . . . 24
sce_mpnst_example . . . 25
sce_overlay_example . . . 26
scOverlay . . . 27

Index **28**

`arrangeNestedPlots` *Arrange plots into a nested patchwork layout*

Description

Internal helper used by `plotOverlayPerNestedGroup` to arrange a named list of plots in rows defined by an outer grouping variable. Each row contains inner groups ordered according to `group_order` when provided. Compact layout skips missing outer/inner combinations and right-pads shorter rows with blank spacers so panel widths stay aligned. Grid layout inserts blank spacers for missing combinations in a full rectangular grid.

Usage

```
arrangeNestedPlots(
  sce,
  plots,
  group_col,
  outer_col,
  group_order = NULL,
  outer_order = NULL,
  spacer = NULL,
  layout = c("compact", "grid")
)
```

Arguments

`sce`	A SingleCellExperiment object.
`plots`	A named list of **ggplot2** or **patchwork** plots. Names should correspond to inner group values, or preferably to nested keys of the form `outer__group`.
`group_col`	A single character value with the name of the `colData(sce)` column defining the inner groups.
`outer_col`	A single character value with the name of the `colData(sce)` column defining the outer groups.

group_order	Optional character vector specifying the order of the inner groups. Defaults to the order observed in group_col.
outer_order	Optional character vector specifying the order of the outer groups. Defaults to the order observed in outer_col.
spacer	Ignored. Kept only for internal compatibility.
layout	Character value controlling the nested arrangement. Accepted values are "compact" and "grid".

Value

A **patchwork** object with one row per outer group and a consistent number of columns across rows.

---

getPalette *Resolve a scOverlay palette*

Description

getPalette converts a palette specification into a vector of colours that can be used by scOverlay or directly in a **ggplot2** scale.

Usage

```
getPalette(
  palette = NULL,
  type = c("continuous", "categorical"),
  n = NULL,
  values = NULL
)
```

Arguments

palette	Palette specification. It can be NULL, a single palette name, a vector of colours or a palette function. A palette function should take the number of requested colours as its first argument and return a colour vector. Defaults to NULL.
type	Character value indicating the type of palette to return. Accepted values are "continuous" and "categorical". Defaults to "continuous".
n	Number of colours to return. If NULL, it is inferred from values when available, otherwise a default of 256 is used. Non-integer values of n are truncated with as.integer(). Defaults to NULL.
values	Optional character vector with the expected categorical values. When supplied and the resolved palette is named, names are used to map colours to values. Defaults to NULL.

Details

Palette specifications can be NULL, the name of one of the palettes bundled with scOverlay, a vector of colours, or a palette function. Palette names are first looked up in scOverlay's internal palette registry for the requested type, then in **viridisLite**, and then among ColorBrewer palettes through **scales**.

When `palette = NULL`, `getPalette` uses the default scOverlay palette for the requested type: `"scOverlay"` for categorical palettes and `"gray_red"` for continuous palettes.

Continuous palettes are represented as anchor colours and interpolated internally to the requested number of colours. This means that a user supplied vector such as `c("gray90", "red", "darkred")` can be used as a continuous gradient.

Categorical palettes can be named. When `values` is supplied and all values are present in the palette names, colours are matched by name. When no values are present in the palette names, colours are assigned by position. If only some values match, `getPalette()` raises an error to avoid silent colour remapping. Built-in categorical palettes intentionally keep integer-like names such as `"1"`, `"2"` and `"3"` because these are useful for cluster labels. Use `unname()` to force positional matching for a named palette, or `setNames()` to assign or replace names.

If a categorical palette has fewer colours than requested, additional colours are generated by interpolation. The categorical palette `"combinationBright"` contains 32 colours and is used by default for `fg_type = "combination"` foregrounds in `plotOverlay`.

Value

A character vector of colours. For categorical palettes, the returned vector is named with `values` when `values` is supplied.

See Also

`listPalettes`, `plotOverlay`.

Examples

```
getPalette("gray_red", type = "continuous", n = 5)

getPalette(
    palette = c("#E6E6E6", "#EF6F6C", "#67000D"),
    type = "continuous",
    n = 5
)

getPalette(
    palette = "scOverlay",
    type = "categorical",
    n = 3,
    values = c("A", "B", "C")
)

getPalette("combinationBright", type = "categorical", n = 8)

getPalette(
    palette = c(A = "red", B = "blue"),
    type = "categorical",
    values = c("B", "A")
)

pal <- getPalette("scOverlay", type = "categorical", n = 4)
setNames(unname(pal), c("Nerve", "PNF", "ANF", "MPNST"))
```

`listPalettes` *List available scOverlay palettes*

Description

`listPalettes` returns metadata for the palettes bundled with scOverlay.

Usage

```
listPalettes(type = c("all", "categorical", "continuous", "bivariate"))
```

Arguments

`type`	Character value indicating which palettes to list. Accepted values are `"all"`, `"categorical"`, `"continuous"` and `"bivariate"`. Defaults to `"all"`.

Details

Categorical palettes return their stored colour vectors. Internally defined continuous palettes return their stored anchor colours. Externally generated continuous palettes, such as **viridisLite** palettes, return `n = NA_integer_` and `colours = NA` because scOverlay does not store a fixed colour vector for them. Bivariate palettes return their four named corner colours: `low_xy`, `high_x`, `high_y` and `high_xy`.

The `is_default` column marks the palettes used by `getPalette` when `palette = NULL`: `"scOverlay"` for categorical palettes, `"gray_red"` for continuous palettes and `"bivariateBright"` for bivariate foregrounds.

Value

A `data.frame` with palette `name`, `type`, stored colour count `n`, a `colours` list-column containing stored palette definitions, and a logical `is_default` column indicating the default palettes.

See Also

`getPalette`, `plotPalettes`, `plotBivariateLegend` and `plotOverlay`.

Examples

```
listPalettes()

listPalettes(type = "bivariate")

pal_info <- listPalettes(type = "bivariate")
pal_info$colours[[which(pal_info$name == "bivariateBright")]]

pal_info <- listPalettes(type = "continuous")
pal_info$colours[[which(pal_info$name == "gray_blue")]]
```

plotBivariateLegend *Plot a standalone bivariate colour legend*

Description

plotBivariateLegend creates a square tile legend for bivariate foreground palettes used by plotOverlay when fg_type = "bivariate". It returns a standalone **ggplot2** object that can be combined manually with an overlay plot, for example with **patchwork**.

Usage

```
plotBivariateLegend(features, palette = "bivariateBright", n_bins = 9L)
```

Arguments

features	Character vector of length two. The first feature labels the x direction and maps to the high_x palette corner; the second feature labels the y direction and maps to high_y.
palette	Bivariate palette specification. It can be one of the built-in bivariate palette names, such as "bivariateBright", or a four-colour vector with corners named low_xy, high_x, high_y and high_xy. Unnamed four-colour vectors are interpreted in that order. Defaults to "bivariateBright".
n_bins	Number of bins per axis in the displayed colour grid. Must be a single integer greater than or equal to 2. Defaults to 9.

Value

A **ggplot2** object containing the bivariate colour grid.

Examples

```
data("sce_overlay_example")

p <- plotOverlay(
    sce_overlay_example,
    foreground = c("Gene1", "Gene2"),
    fg_type = "bivariate"
)

leg <- plotBivariateLegend(c("Gene1", "Gene2"))
leg

patchwork::wrap_plots(
    p,
    plotBivariateLegend(features = c("Gene1", "Gene2")),
    widths = c(1, 0.2)
)

custom_pal <- c(
    low_xy = "#e6e6e6",
    high_x = "#ff4346",
    high_y = "#55b2ff",
```

```
    high_xy = "#7a1fa2"
)

plotBivariateLegend(
    features = c("Gene1", "Gene2"),
    palette = custom_pal
)
```

`plotGeneOverlay` *Plot overlays for several genes*

Description

`plotGeneOverlay` creates one `plotOverlay` plot for each gene in `genes` and returns them as a named list.

Usage

```
plotGeneOverlay(sce, genes, ...)
```

Arguments

`sce`	A SingleCellExperiment object containing the reduced dimensions and gene expression values to plot.
`genes`	Character vector with the genes to plot. Genes are looked up in `rownames(sce)`. This argument has no default and must be provided by the user.
`...`	Additional arguments passed to `plotOverlay`, such as `reduced_dim`, `background`, `bg_type`, `bg_assay`, `fg_assay`, `fg_order`, palettes or rasterization options.

Details

This is a convenience function to quickly inspect the expression pattern of several genes on the same embedding. All additional plotting arguments are passed to `plotOverlay`.

Genes not present in `rownames(sce)` are skipped with a warning. If none of the requested genes are found, the function stops with an error.

Value

A named list of **ggplot2** objects, one for each gene found in `rownames(sce)`.

See Also

`plotOverlay`, `getPalette` and `listPalettes`.

Examples

```
data("sce_overlay_example")

plots <- plotGeneOverlay(
    sce = sce_overlay_example,
    genes = c("SOX10", "S100B"),
    reduced_dim = "TSNE",
    background = "cluster",
    fg_order = "ascending"
)

plots$SOX10
```

---

plotOverlay *Plot layered information on a single-cell embedding*

Description

plotOverlay creates a layered plot of a reduced dimension representation stored in a SingleCellExperiment object. It first creates a background layer, dims this background with a semi-transparent white overlay, and finally draws a foreground variable on top using an independent colour scale. When a layer is set to "none", the corresponding point layer is kept in the plot structure without drawing visible points.

Usage

```
plotOverlay(
  sce,
  foreground,
  background = "solid",
  reduced_dim = "TSNE",
  fg_subset_cells = NULL,
  bg_type = c("auto", "continuous", "categorical"),
  fg_type = c("auto", "continuous", "categorical", "bivariate", "combination"),
  bg_assay = "logcounts",
  fg_assay = "logcounts",
  fg_threshold = 0,
  fg_order = c("auto", "input", "random", "ascending", "descending"),
  bg_point_size = 4,
  fg_point_size = 1,
  bg_palette = NULL,
  fg_palette = NULL,
  bg_raster = FALSE,
  fg_raster = FALSE,
  raster_dpi = 300,
  bg_dimming = 0.9,
  fg_limits = NULL,
  bg_limits = NULL,
  bg_legend = TRUE,
  fg_legend = TRUE,
```

```
  bg_legend_title = NULL,
  fg_legend_title = NULL,
  title = NULL,
  bg_values = NULL,
  fg_values = NULL
)
```

Arguments

sce	A SingleCellExperiment object containing the reduced dimensions and the data to plot. The object must contain the reduced dimension specified in reduced_dim. If background or foreground is a feature, the object must also contain the corresponding assay specified in bg_assay or fg_assay.
foreground	A character value specifying the variable to draw in the foreground layer. It can be "solid" to draw selected foreground cells with a fixed colour, "none" to draw no visible foreground points, a feature in rownames(sce), in which case values are taken from fg_assay, or a column in colData(sce). When fg_type = "bivariate", it must be a character vector of exactly two numeric features or colData(sce) columns. When fg_type = "combination", it must contain between 2 and 5 numeric features or colData(sce) columns. This argument has no default and must be provided by the user.
background	A single character value specifying the background layer. It can be "solid" to draw all background cells with a fixed colour, "none" to draw no visible background points, a feature in rownames(sce), or a column in colData(sce). Defaults to "solid".
reduced_dim	A single character value with the name of the reduced dimension to plot. It must be one of reducedDimNames(sce). Only the first two dimensions are used. Defaults to "TSNE".
fg_subset_cells	A cell subset defining which cells are shown in the foreground layer. It can be NULL, a logical vector of length ncol(sce), a character vector with cell names, a function receiving sce and returning a logical vector, or an expression evaluated in colData(sce). NULL selects all cells. NA values are treated as FALSE. If no cells match, an empty foreground layer is drawn and a warning is emitted. Defaults to NULL.
bg_type	A character value indicating how the background variable should be represented when background is not "solid" or "none". Accepted values are "auto", "continuous" and "categorical". With "auto", feature values are treated as continuous and colData variables are treated as continuous only when they are numeric. Defaults to "auto".
fg_type	A character value indicating how the foreground variable should be represented when foreground is not "solid" or "none". Accepted values are "auto", "continuous", "categorical", "bivariate" and "combination". With "auto", feature values are treated as continuous and colData variables are treated as continuous only when they are numeric. "bivariate" blends colours from two continuous foreground variables. "combination" calls each of 2 to 5 numeric foreground variables expressed when its value is greater than fg_threshold, then treats the expressed/not-expressed combinations as categorical values. Defaults to "auto".
bg_assay	A single character value with the name of the assay to use when background is a feature in rownames(sce). It is ignored when background = "solid" or background = "none". Defaults to "logcounts".

fg_assay
: A single character value with the name of the assay to use when foreground is a feature in rownames(sce). It is ignored when foreground = "solid" or foreground = "none". Defaults to "logcounts".

fg_threshold
: Thresholds used only when fg_type = "combination". It can be a single numeric value, a numeric vector with one threshold per foreground variable, a named numeric vector containing all foreground variable names, or the single character value "median". Values are considered expressed when value > threshold; with the default 0, exact zero values are not expressed. Defaults to 0.

fg_order
: A character value controlling the drawing order of foreground cells. Accepted values are "auto", "input", "random", "ascending" and "descending". "auto" uses "ascending" for bivariate and combination foregrounds and "input" for other foreground types. "input" keeps the current order, "random" randomly shuffles foreground cells, "ascending" draws lower foreground values first and higher values on top, and "descending" does the opposite. For bivariate foregrounds, ascending and descending order by the sum of the two scaled foreground values. For combination foregrounds, ascending and descending order by the deterministic combination level, from all-negative to all-positive. For reproducible random ordering, call set.seed() before plotOverlay(). Defaults to "auto".

bg_point_size
: Numeric value with the point size of the background layer. Defaults to 4.

fg_point_size
: Numeric value with the point size of the foreground layer. Defaults to 1.

bg_palette
: Palette specification for the background layer. It is ignored when background = "none". When background = "solid", NULL uses "gray70" and a single valid colour can be used to set the solid background colour. For variable backgrounds, it can be NULL, a palette name, a vector of colours or a palette function, and the value is passed to getPalette. Defaults to NULL, which uses "gray_red" for continuous backgrounds and "scOverlay" for categorical backgrounds.

fg_palette
: Palette specification for the foreground layer. It is ignored when foreground = "none". When foreground = "solid", NULL uses "red" and a single valid colour can be used to set the solid foreground colour. For variable foregrounds, it can be NULL, a palette name, a vector of colours or a palette function, and the value is passed to getPalette. For continuous foregrounds, colour vectors are interpreted as gradient anchors and interpolated internally. For bivariate foregrounds, NULL uses "bivariateBright" and users can provide a built-in bivariate palette name or a four-colour vector with corners named low_xy, high_x, high_y and high_xy. The first foreground variable maps to the x direction and high_x; the second maps to the y direction and high_y. For combination foregrounds, NULL uses "combinationBright"; user-supplied palettes still override this default. Defaults to NULL, which uses "gray_red" for continuous foregrounds and "scOverlay" for ordinary categorical foregrounds.

bg_raster
: Logical value. If TRUE, rasterize the background point layer using ggrastr::geom_point_rast(). Defaults to FALSE.

fg_raster
: Logical value. If TRUE, rasterize the foreground point layer using ggrastr::geom_point_rast(). Defaults to FALSE.

raster_dpi
: Positive numeric value with the resolution, in dots per inch, used for rasterized point layers. Defaults to 300.

bg_dimming
: Numeric value between 0 and 1 controlling the opacity of the white dimming layer placed between background and foreground. A value of 0 does not dim

the background and a value of 1 completely covers it with white. Defaults to 0.9.

fg_limits — Limits for the foreground colour scale when the foreground is continuous. It can be NULL, a strictly increasing numeric vector of length two, or "shared". For fg_type = "bivariate", NULL uses automatic scaling and "shared" can be used by grouped plotting helpers for comparable bivariate colours across panels. Numeric foreground limits are not supported for bivariate colours. For fg_type = "combination", NULL and "shared" are accepted but numeric limits are not supported. It is ignored with a warning when the foreground layer is not continuous, bivariate or combination. Defaults to NULL.

bg_limits — Limits for the background colour scale when the background is continuous. It can be NULL, a strictly increasing numeric vector of length two, or "shared". In a single plot, "shared" resolves to the range of background values in that plot. It is ignored with a warning when the background layer is not continuous. Defaults to NULL.

bg_legend — Logical value indicating whether to show the background legend. It is ignored when background = "solid" or background = "none". Defaults to TRUE.

fg_legend — Logical value indicating whether to show the foreground legend. It is ignored when foreground = "solid" or foreground = "none". Bivariate foregrounds do not use a normal ggplot foreground legend; use plotBivariateLegend to create a separate bivariate legend. Defaults to TRUE.

bg_legend_title — Character value with the title of the background legend. If NULL, background is used. Defaults to NULL.

fg_legend_title — Character value with the title of the foreground legend. If NULL, foreground is used. For combination plots, two foreground variables are joined with " / ", while larger combinations use "Gene combination". Defaults to NULL.

title — Character value with the plot title. If NULL, foreground is used, except when foreground = "none", where an empty title is used. Defaults to NULL.

bg_values — Categorical value set for the background scale. This is used by grouped plotting helpers to keep categorical colours consistent across panels. Not intended for users.

fg_values — Categorical value set for the foreground scale. This is used by grouped plotting helpers to keep categorical colours consistent across panels. Not intended for users.

Details

This representation is useful to highlight gene expression, scores, annotations or selected groups of cells while keeping the complete embedding visible as context.

The function uses two visual layers. The background layer is used to show the global structure of the embedding, usually coloured by a cell annotation such as a cluster, sample or cell type. Either layer can also be set to "none" to keep the layer position without drawing visible points. The foreground layer is then drawn on top and can represent a fixed-colour selection, a feature from one of the assays, typically a gene expression value, or a variable stored in colData(sce).

The foreground can be restricted to a subset of cells using fg_subset_cells. This only affects the foreground layer: cells not selected by fg_subset_cells are still shown in the background. This is the main idea of the overlay plot and allows highlighting a particular group of cells without losing the context of the whole dataset.

The foreground colour scale is independent from the background colour scale. Palettes can be specified by name, as a vector of colours, or as a palette function. See `getPalette` and `listPalettes` for the available palettes and palette handling rules. For continuous palettes, colour vectors are used as anchors and interpolated internally.

For dense embeddings, the background and foreground point layers can be rasterized independently with `bg_raster` and `fg_raster`. Only the point layers are rasterized; axes, labels, legends and other plot elements remain vector graphics.

Value

A **ggplot2** object with the background, dimming and foreground layers.

See Also

`plotGeneOverlay`, `plotOverlayPerGroup`, `plotOverlayPerNestedGroup`, `getPalette` and `listPalettes`.

Examples

```
data("sce_overlay_example")

# Overlay the expression of a gene on top of the cluster annotation.
p1 <- plotOverlay(
    sce = sce_overlay_example,
    reduced_dim = "TSNE",
    background = "cluster",
    foreground = "SOX10"
)
p1

# Use a solid background and draw higher expression values on top.
p2 <- plotOverlay(
    sce = sce_overlay_example,
    reduced_dim = "TSNE",
    background = "solid",
    foreground = "S100B",
    fg_order = "ascending"
)
p2

# Restrict the foreground to one sample type while keeping all cells in the
# background.
p3 <- plotOverlay(
    sce = sce_overlay_example,
    reduced_dim = "UMAP",
    background = "cluster",
    foreground = "SOX10",
    fg_subset_cells = quote(sample_type == "ANNUBP")
)
p3

# Use a custom continuous palette and rasterize the background point layer.
p4 <- plotOverlay(
    sce = sce_overlay_example,
    reduced_dim = "TSNE",
    background = "sample_type",
    foreground = "SOX10",
```

```
    fg_palette = c("#E6E6E6", "#EF6F6C", "#67000D"),
    bg_raster = TRUE
)
p4

# Blend two numeric foreground features with a bivariate palette.
p5 <- plotOverlay(
    sce_overlay_example,
    foreground = c("Gene1", "Gene2"),
    fg_type = "bivariate",
    fg_palette = "bivariateBright"
)
p5

# Create a separate bivariate legend.
leg <- plotBivariateLegend(
    features = c("Gene1", "Gene2"),
    palette = "bivariateBright"
)
leg

# Call expression combinations from two numeric foreground features.
p6 <- plotOverlay(
    sce_overlay_example,
    foreground = c("Gene1", "Gene2"),
    fg_type = "combination",
    fg_threshold = 0
)
p6

# Highlight a subset with a solid foreground colour.
p7 <- plotOverlay(
    sce_overlay_example,
    foreground = "solid",
    background = "cluster",
    fg_subset_cells = quote(cluster == "C1")
)
p7
```

`plotOverlayPerGroup` *Plot overlays split by a cell metadata group*

Description

`plotOverlayPerGroup` creates one `plotOverlay` plot for each group defined by a column in `colData(sce)` and combines the plots into a single **patchwork** object.

Usage

```
plotOverlayPerGroup(
  sce,
  group_col,
  foreground,
```

```
  background = "solid",
  reduced_dim = "TSNE",
  group_order = NULL,
  fg_subset_cells = NULL,
  bg_type = c("auto", "continuous", "categorical"),
  bg_assay = "logcounts",
  fg_assay = "logcounts",
  fg_type = c("auto", "continuous", "categorical", "bivariate", "combination"),
  fg_threshold = 0,
  fg_order = c("auto", "input", "random", "ascending", "descending"),
  bg_point_size = 2,
  fg_point_size = 0.5,
  bg_palette = NULL,
  fg_palette = NULL,
  bg_raster = FALSE,
  fg_raster = FALSE,
  raster_dpi = 300,
  bg_dimming = 0.6,
  fg_limits = NULL,
  bg_limits = NULL,
  bg_legend = TRUE,
  fg_legend = TRUE,
  bg_legend_title = NULL,
  fg_legend_title = NULL,
  shared_legend = FALSE
)
```

Arguments

sce	A SingleCellExperiment object containing the reduced dimensions and the data to plot. The object must contain the reduced dimension specified in `reduced_dim`, the grouping column specified in `group_col`, and the data required by `foreground`.
group_col	A single character value with the name of the `colData(sce)` column used to split the plot into panels. Each value in this column defines one panel. This argument has no default and must be provided by the user.
foreground	Foreground layer specification. It can be a feature name in `rownames(sce)`, a `colData(sce)` column, `"solid"`, `"none"`, or a character vector of feature or column names for `fg_type = "bivariate"` or `fg_type = "combination"`. This argument has no default and must be provided by the user.
background	A single character value specifying the background layer passed to `plotOverlay`. It can be `"solid"` to draw all background cells with a fixed colour, `"none"` to draw no visible background points, a feature in `rownames(sce)`, or a column in `colData(sce)`. Defaults to `"solid"`.
reduced_dim	A single character value with the name of the reduced dimension to plot. It must be one of `reducedDimNames(sce)`. Only the first two dimensions are used. Defaults to `"TSNE"`.
group_order	Optional character vector specifying the order in which groups should be plotted. If `NULL`, factor levels are used for factor grouping variables; otherwise, the order of appearance in `group_col` is used. Defaults to `NULL`.
fg_subset_cells	An optional additional cell subset applied to the foreground layer before splitting by group. It can be `NULL`, a logical vector of length `ncol(sce)`, a character

vector with cell names, a function receiving sce and returning a logical vector, or an expression evaluated in colData(sce). NULL selects all cells. NA values are treated as FALSE. Defaults to NULL. Groups with no matching foreground cells still produce panels with empty foreground layers.

bg_type
: A character value indicating how the background variable should be represented when background is not "solid" or "none". Accepted values are "auto", "continuous" and "categorical". See plotOverlay for details. Defaults to "auto".

bg_assay
: A single character value with the name of the assay to use when background is a feature in rownames(sce). It is ignored when background = "solid" or background = "none". Defaults to "logcounts".

fg_assay
: A single character value with the name of the assay to use when foreground is a feature in rownames(sce). It is ignored when foreground = "solid" or foreground = "none". Defaults to "logcounts".

fg_type
: A character value indicating how the foreground variable should be represented when foreground is not "solid" or "none". Accepted values are "auto", "continuous", "categorical", "bivariate" and "combination". With "auto", feature values are treated as continuous and colData variables are treated as continuous only when they are numeric. "bivariate" blends colours from exactly two numeric foreground variables. "combination" calls 2 to 5 numeric foreground variables expressed using fg_threshold and treats the resulting combinations as categorical values. Defaults to "auto".

fg_threshold
: Thresholds used only when fg_type = "combination". It can be a single numeric value, a numeric vector with one threshold per foreground variable, a named numeric vector containing all foreground variable names, or "median". Values are expressed when value > threshold; with the default 0, exact zero values are not expressed. Defaults to 0.

fg_order
: A character value controlling the drawing order of foreground cells within each panel. Accepted values are "auto", "input", "random", "ascending" and "descending". "auto" uses "ascending" for bivariate and combination foregrounds and "input" for other foreground types. For bivariate foregrounds, ascending and descending order by the sum of the two scaled foreground values. For combination foregrounds, ascending and descending order by deterministic combination level, from all-negative to all-positive. See plotOverlay for details. Defaults to "auto".

bg_point_size
: Numeric value with the point size of the background layer in each panel. Defaults to 2.

fg_point_size
: Numeric value with the point size of the foreground layer in each panel. Defaults to 0.5.

bg_palette
: Palette specification for the background layer. It can be NULL, a palette name, a vector of colours or a palette function. When background = "solid", NULL uses "gray70" and a single valid colour can be used to set the solid background colour. It is ignored when background = "none". For variable backgrounds, the value is passed to getPalette. Defaults to NULL, which uses "gray_red" for continuous backgrounds and "scOverlay" for categorical backgrounds.

fg_palette
: Palette specification for the foreground layer. When foreground = "solid", NULL uses "red" and a single valid colour can be used to set the solid foreground colour. It is ignored when foreground = "none". For variable foregrounds, it can be NULL, a palette name, a vector of colours or a palette function. The value

is passed to getPalette. For continuous foregrounds, colour vectors are interpreted as gradient anchors and interpolated internally. For bivariate foregrounds, NULL uses "bivariateBright" and users can provide a built-in bivariate palette name or a four-colour vector with corners named low_xy, high_x, high_y and high_xy. For combination foregrounds, NULL uses "combinationBright"; user-supplied palettes still override this default. Defaults to NULL, which uses "gray_red" for continuous foregrounds and "scOverlay" for ordinary categorical foregrounds.

bg_raster — Logical value. If TRUE, rasterize the background point layer in each panel using ggrastr::geom_point_rast(). Defaults to FALSE.

fg_raster — Logical value. If TRUE, rasterize the foreground point layer in each panel using ggrastr::geom_point_rast(). Defaults to FALSE.

raster_dpi — Positive numeric value with the resolution, in dots per inch, used for rasterized point layers. Defaults to 300.

bg_dimming — Numeric value between 0 and 1 controlling the opacity of the white dimming layer placed between background and foreground in each panel. A value of 0 does not dim the background and a value of 1 completely covers it with white. Defaults to 0.6.

fg_limits — Limits for the foreground colour scale when the foreground is continuous. It can be NULL, a strictly increasing numeric vector of length two, or "shared". For fg_type = "bivariate", NULL uses automatic per-panel scaling and "shared" computes separate shared ranges for the two numeric foreground variables. Numeric foreground limits are not supported for bivariate colours. For fg_type = "combination", NULL and "shared" are accepted but numeric limits are not supported. Defaults to NULL.

bg_limits — Limits for the background colour scale when the background is continuous. It can be NULL, a strictly increasing numeric vector of length two, or "shared". With "shared", one range is computed over all cells and reused in every panel. Defaults to NULL.

bg_legend — Logical value indicating whether to show the background legend in each panel. It is ignored when background = "solid". Defaults to TRUE.

fg_legend — Logical value indicating whether to show the foreground legend in each panel. It is ignored when foreground = "solid". Defaults to TRUE. Bivariate foregrounds do not use a normal ggplot foreground legend; use plotBivariateLegend to create a separate bivariate legend.

bg_legend_title — Character value with the title of the background legend. If NULL, background is used. Defaults to NULL.

fg_legend_title — Character value with the title of the foreground legend. If NULL, foreground is used. For combination plots, two foreground variables are joined with " / ", while larger combinations use "Gene combination". Defaults to NULL.

shared_legend — Logical value indicating whether compatible legends should be collected into a shared patchwork legend. Defaults to FALSE.

Details

This function is useful to compare the same foreground variable across samples, sample types, clusters or any other cell-level annotation. Each panel keeps the full embedding as background, while the foreground layer is restricted to the cells belonging to the corresponding group.

The grouping variable is specified with `group_col`. For each value in this column, `plotOverlayPerGroup` calls `plotOverlay` using the same background, foreground, palette and plotting options. The foreground subset for each panel is defined by the group membership and, optionally, by the additional filter supplied with `fg_subset_cells`.

As in `plotOverlay`, subsetting only affects the foreground layer. Unless `background = "none"`, the background layer still shows all cells in every panel, providing a common reference embedding across groups. This makes it possible to compare where cells from different samples or sample types are located in the same reduced dimension space.

With `fg_limits = "shared"`, foreground ranges are computed after the global `fg_subset_cells` filter and before splitting into groups. In bivariate mode, separate shared ranges are computed for the two numeric foreground variables so colours are comparable across panels. With `bg_limits = "shared"`, one background range is computed across all cells. Categorical foreground and background palettes are based on global categorical values, so colours remain consistent across panels. Named categorical palettes are recommended when strict category-colour control is needed.

If `shared_legend = TRUE`, compatible legends are collected with **patchwork**. The legend can be repositioned afterwards with patchwork syntax, for example `p & ggplot2::theme(legend.position = "bottom")`.

If `group_order` is provided, panels are drawn following that order. If it is `NULL`, factor levels are used when `group_col` is a factor; otherwise, groups are shown in their order of appearance in the object.

Value

A **patchwork** object combining one `plotOverlay` plot per group. If no valid groups produce a panel, the function returns `NULL` with a warning.

See Also

`plotOverlay`, `plotOverlayPerNestedGroup`, `plotGeneOverlay`, `getPalette` and `listPalettes`.

Examples

```
data("sce_overlay_example")

# Plot SOX10 expression separately for each sample type.
p1 <- plotOverlayPerGroup(
    sce = sce_overlay_example,
    foreground = "SOX10",
    group_col = "sample_type",
    reduced_dim = "TSNE",
    background = "solid",
    fg_order = "ascending"
)
p1

# Use the same grouping but colour the background by cluster.
p2 <- plotOverlayPerGroup(
    sce = sce_overlay_example,
    foreground = "S100B",
    group_col = "sample_type",
    reduced_dim = "TSNE",
    background = "cluster",
    fg_order = "ascending"
)
```

```
p2

# Specify an explicit group order.
p3 <- plotOverlayPerGroup(
    sce = sce_overlay_example,
    foreground = "SOX10",
    group_col = "sample_type",
    group_order = c("PNF", "ANNUBP", "MPNST"),
    reduced_dim = "TSNE",
    background = "solid"
)
p3

# Compare expression across samples with shared foreground limits and a
# collected legend.
p4 <- plotOverlayPerGroup(
    sce = sce_overlay_example,
    group_col = "sample",
    foreground = "Gene1",
    background = "cluster",
    fg_limits = "shared",
    shared_legend = TRUE
)
p4

# Use shared bivariate scaling across sample panels.
p5 <- plotOverlayPerGroup(
    sce = sce_overlay_example,
    group_col = "sample",
    foreground = c("Gene1", "Gene2"),
    fg_type = "bivariate",
    fg_limits = "shared"
)
p5
```

`plotOverlayPerNestedGroup` *Plot overlays arranged by nested cell metadata groups*

Description

`plotOverlayPerNestedGroup` creates one `plotOverlay` plot for each observed combination of an inner and outer grouping variable. The usual use case is to create per-sample overlay plots and arrange them by sample type, condition or experimental group.

Usage

```
plotOverlayPerNestedGroup(
  sce,
  group_col,
  outer_col,
  foreground,
  background = "solid",
```

```
  reduced_dim = "TSNE",
  group_order = NULL,
  outer_order = NULL,
  layout = c("auto", "compact", "grid"),
  panel_titles = c("auto", "outer_inner", "inner", "none"),
  fg_subset_cells = NULL,
  bg_type = c("auto", "continuous", "categorical"),
  bg_assay = "logcounts",
  fg_assay = "logcounts",
  fg_type = c("auto", "continuous", "categorical", "bivariate", "combination"),
  fg_threshold = 0,
  fg_order = c("auto", "input", "random", "ascending", "descending"),
  bg_point_size = 2,
  fg_point_size = 0.5,
  bg_palette = NULL,
  fg_palette = NULL,
  bg_raster = FALSE,
  fg_raster = FALSE,
  raster_dpi = 300,
  bg_dimming = 0.6,
  fg_limits = NULL,
  bg_limits = NULL,
  bg_legend = TRUE,
  fg_legend = TRUE,
  bg_legend_title = NULL,
  fg_legend_title = NULL,
  shared_legend = FALSE
)
```

Arguments

sce	A SingleCellExperiment object containing the reduced dimensions and the data to plot. The object must contain the reduced dimension specified in `reduced_dim`, the grouping columns specified in `group_col` and `outer_col`, and the data required by `foreground`.
group_col	A single character value with the name of the `colData(sce)` column used to define the inner groups, usually the panels. This argument has no default and must be provided by the user.
outer_col	A single character value with the name of the `colData(sce)` column used to arrange the inner groups, usually as rows in the final plot. This argument has no default and must be provided by the user.
foreground	Foreground layer specification. It can be a feature name in `rownames(sce)`, a `colData(sce)` column, `"solid"`, `"none"`, or a character vector of feature or column names for `fg_type = "bivariate"` or `fg_type = "combination"`. This argument has no default and must be provided by the user.
background	A single character value specifying the background layer passed to `plotOverlay`. It can be `"solid"` to draw all background cells with a fixed colour, `"none"` to draw no visible background points, a feature in `rownames(sce)`, or a column in `colData(sce)`. Defaults to `"solid"`.
reduced_dim	A single character value with the name of the reduced dimension to plot. It must be one of `reducedDimNames(sce)`. Only the first two dimensions are used. Defaults to `"TSNE"`.

group_order — Optional character vector specifying the order in which inner groups should be plotted. If NULL, factor levels are used for factor grouping variables; otherwise, the order of appearance in group_col is used. Defaults to NULL.

outer_order — Optional character vector specifying the order of the outer groups. If NULL, factor levels are used for factor grouping variables; otherwise, the order of appearance in outer_col is used. Defaults to NULL.

layout — Character value controlling how nested panels are arranged. Use "compact" to show only observed group_col values within each outer_col row, "grid" to reserve a full outer_col-by-group_col grid with blank spacers for missing combinations, or "auto" to use "grid" when any group_col value appears in more than one outer_col value and "compact" otherwise. Defaults to "auto".

panel_titles — Character value controlling titles added to real nested panels. Use "outer_inner" to label panels as inner - outer, "inner" to label panels by the inner group only, "none" to suppress panel titles, or "auto" to use "outer_inner". Defaults to "auto".

fg_subset_cells — An optional additional cell subset applied to the foreground layer before splitting by group. It can be NULL, a logical vector of length ncol(sce), a character vector with cell names, a function receiving sce and returning a logical vector, or an expression evaluated in colData(sce). NULL selects all cells. NA values are treated as FALSE. Defaults to NULL. Inner groups with no matching foreground cells still produce panels with empty foreground layers, preserving the nested layout.

bg_type — A character value indicating how the background variable should be represented when background is not "solid" or "none". Accepted values are "auto", "continuous" and "categorical". See plotOverlay for details. Defaults to "auto".

bg_assay — A single character value with the name of the assay to use when background is a feature in rownames(sce). It is ignored when background = "solid" or background = "none". Defaults to "logcounts".

fg_assay — A single character value with the name of the assay to use when foreground is a feature in rownames(sce). It is ignored when foreground = "solid" or foreground = "none". Defaults to "logcounts".

fg_type — A character value indicating how the foreground variable should be represented when foreground is not "solid" or "none". Accepted values are "auto", "continuous", "categorical", "bivariate" and "combination". With "auto", feature values are treated as continuous and colData variables are treated as continuous only when they are numeric. "bivariate" blends colours from exactly two numeric foreground variables. "combination" calls 2 to 5 numeric foreground variables expressed using fg_threshold and treats the resulting combinations as categorical values. Defaults to "auto".

fg_threshold — Thresholds used only when fg_type = "combination". It can be a single numeric value, a numeric vector with one threshold per foreground variable, a named numeric vector containing all foreground variable names, or "median". Values are expressed when value > threshold; with the default 0, exact zero values are not expressed. Defaults to 0.

fg_order — A character value controlling the drawing order of foreground cells within each panel. Accepted values are "auto", "input", "random", "ascending" and "descending". "auto" uses "ascending" for bivariate and combination foregrounds and "input" for other foreground types. For bivariate foregrounds, ascending and descending order by the sum of the two scaled foreground values.

For combination foregrounds, ascending and descending order by deterministic combination level, from all-negative to all-positive. See plotOverlay for details. Defaults to "auto".

bg_point_size — Numeric value with the point size of the background layer in each panel. Defaults to 2.

fg_point_size — Numeric value with the point size of the foreground layer in each panel. Defaults to 0.5.

bg_palette — Palette specification for the background layer. It can be NULL, a palette name, a vector of colours or a palette function. When background = "solid", NULL uses "gray70" and a single valid colour can be used to set the solid background colour. It is ignored when background = "none". For variable backgrounds, the value is passed to getPalette. Defaults to NULL, which uses "gray_red" for continuous backgrounds and "scOverlay" for categorical backgrounds.

fg_palette — Palette specification for the foreground layer. When foreground = "solid", NULL uses "red" and a single valid colour can be used to set the solid foreground colour. It is ignored when foreground = "none". For variable foregrounds, it can be NULL, a palette name, a vector of colours or a palette function. The value is passed to getPalette. For continuous foregrounds, colour vectors are interpreted as gradient anchors and interpolated internally. For bivariate foregrounds, NULL uses "bivariateBright" and users can provide a built-in bivariate palette name or a four-colour vector with corners named low_xy, high_x, high_y and high_xy. For combination foregrounds, NULL uses "combinationBright"; user-supplied palettes still override this default. Defaults to NULL, which uses "gray_red" for continuous foregrounds and "scOverlay" for ordinary categorical foregrounds.

bg_raster — Logical value. If TRUE, rasterize the background point layer in each panel using ggrastr::geom_point_rast(). Defaults to FALSE.

fg_raster — Logical value. If TRUE, rasterize the foreground point layer in each panel using ggrastr::geom_point_rast(). Defaults to FALSE.

raster_dpi — Positive numeric value with the resolution, in dots per inch, used for rasterized point layers. Defaults to 300.

bg_dimming — Numeric value between 0 and 1 controlling the opacity of the white dimming layer placed between background and foreground in each panel. A value of 0 does not dim the background and a value of 1 completely covers it with white. Defaults to 0.6.

fg_limits — Limits for the foreground colour scale when the foreground is continuous. It can be NULL, a strictly increasing numeric vector of length two, or "shared". For fg_type = "bivariate", NULL uses automatic per-panel scaling and "shared" computes separate shared ranges for the two numeric foreground variables. Numeric foreground limits are not supported for bivariate colours. For fg_type = "combination", NULL and "shared" are accepted but numeric limits are not supported. Defaults to NULL.

bg_limits — Limits for the background colour scale when the background is continuous. It can be NULL, a strictly increasing numeric vector of length two, or "shared". With "shared", one range is computed over all cells and reused in every panel. Defaults to NULL.

bg_legend — Logical value indicating whether to show the background legend in each panel. It is ignored when background = "solid". Defaults to TRUE.

fg_legend — Logical value indicating whether to show the foreground legend in each panel. It is ignored when foreground = "solid". Defaults to TRUE. Bivariate foregrounds do not use a normal ggplot foreground legend; use plotBivariateLegend to create a separate bivariate legend.

bg_legend_title — Character value with the title of the background legend. If NULL, background is used. Defaults to NULL.

fg_legend_title — Character value with the title of the foreground legend. If NULL, foreground is used. For combination plots, two foreground variables are joined with " / ", while larger combinations use "Gene combination". Defaults to NULL.

shared_legend — Logical value indicating whether compatible legends should be collected into a shared patchwork legend. Defaults to FALSE.

Details

This function is useful when the cells belong to two related annotations, for example samples nested within sample types. Each panel uses all cells for the background and restricts only the foreground to one observed outer/inner group combination, with an optional additional foreground subset applied inside that panel.

The inner grouping variable is specified with group_col and the outer grouping variable is specified with outer_col. Panels are defined by observed outer_col and group_col combinations, then arranged with one row per outer value. Compact layout shows only observed inner groups within each outer row, then right-pads shorter rows with blank spacers so panel widths remain consistent. It is useful for true nested structures, such as sample type/sample. Grid layout reserves a full outer-by-inner grid and is useful for repeated inner labels, such as sample type/treatment. With layout = "auto", grid layout is used when any inner label appears in more than one outer value; otherwise compact layout is used. Missing combinations in grid layout are blank spacers, not background-only panels. Existing combinations with no foreground-selected cells are still drawn using the existing no-foreground behavior.

For each observed outer/inner combination, plotOverlayPerNestedGroup calls plotOverlay using the same background, foreground, palette and plotting options. The full sce is used in every panel so the background layer always shows the full dataset. The foreground subset for each panel is defined by the inner and outer group membership and, optionally, by the additional filter supplied with fg_subset_cells. Panel titles are added only to real data panels; blank spacer panels do not receive titles.

Missing outer/inner combinations are skipped in compact layout and represented by blank spacers in grid layout. They are not drawn as background-only panels. When a combination exists but fg_subset_cells selects no foreground cells, the panel is still drawn and plotOverlay handles the empty foreground layer in the usual way.

With fg_limits = "shared", foreground ranges are computed after the global fg_subset_cells filter and before splitting into panels. In bivariate mode, separate shared ranges are computed for the two numeric foreground variables so colours are comparable across panels. With bg_limits = "shared", one background range is computed across all cells. Categorical foreground and background palettes are based on global categorical values, so colours remain consistent across panels. Named categorical palettes are recommended when strict category-colour control is needed.

If shared_legend = TRUE, compatible legends are collected with **patchwork**. The legend can be repositioned afterwards with patchwork syntax, for example p & ggplot2::theme(legend.position = "bottom").

The function supports both uniquely nested inner groups and repeated inner labels. For uniquely nested inner groups, such as samples within sample types, layout = "auto" resolves to "compact".

For repeated inner labels, such as treatments or QC groups present across multiple outer groups, `layout = "auto"` resolves to `"grid"`.

If `group_order` or `outer_order` are provided, panels and rows are drawn following those orders. If they are `NULL`, factor levels are used when the corresponding grouping variable is a factor; otherwise, values are shown in their order of appearance in the object.

Value

A **patchwork** object arranging one `plotOverlay` plot per inner group and outer group. If no valid plots can be generated, the function returns `NULL` with a warning.

See Also

`plotOverlay`, `plotOverlayPerGroup`, `plotGeneOverlay`, `getPalette` and `listPalettes`.

Examples

```
data("sce_overlay_example")

# Plot SOX10 expression by sample, arranging samples by sample type.
p1 <- plotOverlayPerNestedGroup(
    sce = sce_overlay_example,
    foreground = "SOX10",
    group_col = "sample",
    outer_col = "sample_type",
    reduced_dim = "TSNE",
    background = "solid",
    fg_order = "ascending"
)
p1

# Use the cluster annotation as background in every panel.
p2 <- plotOverlayPerNestedGroup(
    sce = sce_overlay_example,
    foreground = "S100B",
    group_col = "sample",
    outer_col = "sample_type",
    reduced_dim = "TSNE",
    background = "cluster",
    fg_order = "ascending"
)
p2

# Specify explicit orders for rows and panels.
p3 <- plotOverlayPerNestedGroup(
    sce = sce_overlay_example,
    foreground = "SOX10",
    group_col = "sample",
    outer_col = "sample_type",
    group_order = c("S1", "S2", "S3", "S4", "S5", "S6"),
    outer_order = c("PNF", "ANNUBP", "MPNST"),
    reduced_dim = "TSNE",
    background = "solid",
    layout = "auto"
)
p3
```

```
sce_overlay_example$cluster_group <- ifelse(
    sce_overlay_example$cluster %in% c("C1", "C2"),
    "Cluster C1/C2",
    "Other clusters"
)
p_grid <- plotOverlayPerNestedGroup(
    sce = sce_overlay_example,
    foreground = "Gene1",
    group_col = "cluster_group",
    outer_col = "sample_type",
    reduced_dim = "TSNE",
    background = "cluster",
    fg_limits = "shared",
    shared_legend = TRUE,
    layout = "auto"
)
p_grid

# Reuse one foreground scale and collect compatible legends.
p4 <- plotOverlayPerNestedGroup(
    sce = sce_overlay_example,
    foreground = "Gene1",
    group_col = "sample",
    outer_col = "sample_type",
    reduced_dim = "TSNE",
    background = "cluster",
    fg_limits = "shared",
    shared_legend = TRUE
)
p4
```

`plotPalettes` *Plot available scOverlay palettes*

Description

`plotPalettes` displays the palettes bundled with scOverlay. Categorical and continuous palettes are displayed as one-dimensional swatches. Bivariate palettes are displayed as two-dimensional colour grids using the same orientation as `plotBivariateLegend`: feature 1 increases from left to right and feature 2 increases from bottom to top.

Usage

```
plotPalettes(
  type = c("all", "categorical", "continuous", "bivariate"),
  n = 24,
  show_names = TRUE
)
```

Arguments

type	Character value indicating which palettes to display. Accepted values are "all", "categorical", "continuous" and "bivariate". Defaults to "all".
n	Number of colours used to display continuous palettes, the maximum number of colours shown for categorical palettes, and the number of bins per axis for bivariate palettes. Non-integer values are truncated with as.integer(). Defaults to 24.
show_names	Logical value. If TRUE, categorical palette previews show stored colour names when available, or position labels otherwise. Bivariate palette names are always shown as plot titles. Defaults to TRUE.

Details

Continuous palettes may be sampled or interpolated for display, including externally generated palettes such as **viridisLite** palettes. Use listPalettes to inspect, copy or modify stored palette definitions.

Value

A **ggplot2** object for a single preview section or a **patchwork** object when multiple sections are shown.

See Also

listPalettes, plotBivariateLegend and getPalette.

Examples

```
plotPalettes()
plotPalettes(type = "continuous")
plotPalettes(type = "bivariate")
```

---

sce_mpnst_example *Example MPNST SingleCellExperiment for the scOverlay vignette*

Description

A reduced SingleCellExperiment derived from a single-cell study of peripheral nerve and NF1-associated tumours generated by the package author's research group. The originating study was deposited at the European Genome-phenome Archive (EGAS50000001747).

Format

A SingleCellExperiment with a selected set of log-normalized expression values in the logcounts assay (not raw counts), cell metadata (including sample type, cell types and quality-control metrics), and two-dimensional TSNE coordinates. UMAP and PCA coordinates are retained when available in the processed source object.

Details

The example was extracted from an independently processed `SingleCellExperiment`; the upstream single-cell preprocessing pipeline is not part of scOverlay. Two nerve, two PNF, two ANF and three MPNST samples were selected, with up to 2,500 cells sampled from each. A subset of genes, annotations and embedding coordinates was retained. Original sample IDs were replaced with generic labels (e.g., `Nerve1`, `PNF1`, `ANF1`, `MPNST1`), including the sample prefixes of cell identifiers. No direct patient identifiers are included.

The reduced example data are distributed with scOverlay under the Artistic-2.0 license. This applies to the reduced packaged example and does not change the access conditions of the original EGA study.

Source

Generated by the package author's research group; originating study deposited at the European Genome-phenome Archive, accession EGAS50000001747, https://ega-archive.org/studies/EGAS50000001747.

Examples

```
data("sce_mpnst_example")
sce_mpnst_example
```

`sce_overlay_example` *Small example SingleCellExperiment for scOverlay*

Description

A small SingleCellExperiment object used in scOverlay examples and tests. It contains a small synthetic set of cells, genes, cell annotations and reduced dimensions to demonstrate the plotting functions quickly. It includes `logcounts`, `TSNE` and `UMAP` reduced dimensions, and cell metadata such as `sample`, `sample_type` and `cluster`.

Format

A SingleCellExperiment object with gene expression values, cell metadata and reduced dimensions.

Examples

```
data("sce_overlay_example")
sce_overlay_example
```

scOverlay *scOverlay: layered overlays for single-cell embeddings*

Description

scOverlay creates layered overlay plots for reduced-dimension embeddings stored in SingleCellExperiment objects. It combines background and foreground layers to show gene expression, assay-derived features and `colData` metadata with independent colour scales.

Details

The package includes helpers for single plots, grouped plots and nested grouped plots; bivariate colour blends for two numeric foreground variables; expression-combination plots for small sets of genes or numeric metadata; and palette listing and preview utilities. Background and foreground point layers can be rasterized independently for dense single-cell plots while keeping labels, axes and legends as vector graphics.

Index

∗ **datasets**
sce_mpnst_example, 25
sce_overlay_example, 26
∗ **internal**
arrangeNestedPlots, 2
scOverlay, 27

arrangeNestedPlots, 2

getPalette, 3, *5*, *7*, *10*, *12*, *15–17*, *21*, *23*, *25*

listPalettes, *4*, 5, *7*, *12*, *17*, *23*, *25*

plotBivariateLegend, *5*, 6, *11*, *16*, *22*, *24, 25*
plotGeneOverlay, 7, *12*, *17*, *23*
plotOverlay, *4–7*, 8, *13–15*, *17–23*
plotOverlayPerGroup, *12*, 13, *23*
plotOverlayPerNestedGroup, *2*, *12*, *17*, 18
plotPalettes, *5*, 24

sce_mpnst_example, 25
sce_overlay_example, 26
scOverlay, 27
SingleCellExperiment, *2*, *7–9*, *14*, *19*, *25–27*