Package 'ggsc'

November 12, 2026

Title Visualizing Single Cell and Spatial Transcriptomics

Version 1.10.1

Description Useful functions to visualize single cell and spatial data. It supports visualizing 'Seurat', 'SingleCellExperiment' and 'SpatialExperiment' objects through grammar of graphics syntax implemented in 'ggplot2'.

Depends R (>= 4.1.0)

Imports Rcpp, RcppParallel, cli, dplyr, ggfun (>= 0.1.5), ggplot2, grDevices, grid, methods, rlang, scattermore, stats, Seurat, SingleCellExperiment, SummarizedExperiment, tidydr, tidyr, tibble, utils, RColorBrewer, yulab.utils, scales

Suggests aplot, BiocParallel, forcats, ggforce, ggnewscale, igraph, knitr, ks, Matrix, prettydoc, rmarkdown, scran, scater, scatterpie (>= 0.2.4), scuttle, shadowtext, sf, SeuratObject, SpatialExperiment, STexampleData, testthat (>= 3.0.0), MASS

BugReports https://github.com/YuLab-SMU/ggsc/issues

URL https://github.com/YuLab-SMU/ggsc (devel), https://yulab-smu.top/ggsc/ (docs)

biocViews DimensionReduction, GeneExpression, SingleCell, Software, Spatial, Transcriptomics,Visualization

VignetteBuilder knitr

SystemRequirements GNU make

ByteCompile true

License Artistic-2.0

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Remotes YuLab-SMU/SVP

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

git_branch RELEASE_3_23

git_last_commit d30c9bb

git_last_commit_date 2026-05-17

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Guangchuang Yu [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-6485-8781>), Shuangbin Xu [aut] (ORCID: <https://orcid.org/0000-0003-3513-5362>), Noriaki Sato [ctb]

Maintainer Guangchuang Yu <guangchuangyu@gmail.com>

Contents

ggsc-package . . . 3
CalWkdeCpp . . . 3
draw_key_bgpoint . . . 4
geom_bgpoint . . . 5
geom_scattermore2 . . . 7
plot_lisa_feature . . . 9
reexports . . . 11
scale_bg_color_manual . . . 12
sc_dim . . . 14
sc_dim_count . . . 16
sc_dim_geom_ellipse . . . 17
sc_dim_geom_feature . . . 18
sc_dim_geom_label . . . 19
sc_dim_geom_sub . . . 20
sc_dim_sub . . . 21
sc_dot . . . 22
sc_feature . . . 24
sc_geom_annot . . . 27
sc_geom_point . . . 28
sc_spatial . . . 29
sc_violin . . . 32

Index **35**

`ggsc-package` *ggsc: Visualizing Single Cell and Spatial Transcriptomics*

Description

Useful functions to visualize single cell and spatial data. It supports visualizing 'Seurat', 'SingleCellExperiment' and 'SpatialExperiment' objects through grammar of graphics syntax implemented in 'ggplot2'.

Author(s)

Maintainer: Guangchuang Yu <guangchuangyu@gmail.com> (ORCID) [copyright holder]

Authors:

- Shuangbin Xu <xshuangbin@163.com> (ORCID)

Other contributors:

- Noriaki Sato <nori@hgc.jp> [contributor]

See Also

Useful links:

- https://github.com/YuLab-SMU/ggsc(devel)
- https://yulab-smu.top/ggsc/(docs)
- Report bugs at https://github.com/YuLab-SMU/ggsc/issues

`CalWkdeCpp` *Two-Dimensional Weighted Kernel Density Estimation And Mapping the Result To Original Dimension*

Description

Two-Dimensional Weighted Kernel Density Estimation And Mapping the Result To Original Dimension

Usage

```
CalWkdeCpp(x, w, l, h, adjust = 1, n = 400L)
```

Arguments

x	The 2-D coordinate matrix
w	The weighted sparse matrix, the number columns the same than the number rows than x.
l	The limits of the rectangle covered by the grid as c(xl, xu, yl, yu)
h	The vector of bandwidths for x and y directions, defaults to normal reference bandwidth (see bandwidth.nrd), A scalar value will be taken to apply to both directions (see ks::hpi).
adjust	numeric value to adjust to bandwidth, default is 1.
n	number of grid points in the two directions, default is 400.

---

draw_key_bgpoint *Key drawing functions*

Description

Each Geom has an associated function that draws the key when the geom needs to be displayed in a legend. These are the options built into ggplot2.

Usage

```
draw_key_bgpoint(data, params, size)
```

Arguments

data	A single row data frame containing the scaled aesthetics to display in this key
params	A list of additional parameters supplied to the geom.
size	Width and height of key in mm.

Value

A grid grob.

geom_bgpoint *geom_bgpoint*

Description

this add the background color for geom_point

Usage

```
geom_bgpoint(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  gap_colour = "white",
  gap_alpha = 1,
  bg_line_width = 0.3,
  gap_line_width = 0.1,
  pointsize = NULL
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount.

- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `layer`.

`na.rm` If `FALSE`, the default, missing values are removed with a warning, if `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

`gap_colour` colour of gap background between the bottom background and top point point layer, default is `white`.

`gap_alpha` numeric the transparency of gap background colour, default is 1.

`bg_line_width` numeric the line width of background point layer, default is `0.3`.

`gap_line_width` numeric the line width of gap between the background and top point point layer, default is `.1`.

`pointsize` numeric the size of point, default is NULL, will use the internal size aesthetics of `geom_bgpoint`

Details

Aesthetics `geom_bgpoint` understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- `x`.
- `y`.
- `colour` the colour of point, default is `black`.
- `bg_colour` the colour of background point, default is `NA`.
- `alpha` the transparency of colour, default is 1.
- `subset` subset the data frame which meet conditions to display.

Author(s)

Shuangbin Xu

Examples

```
library(ggplot2)
ggplot(iris,
 aes(x= Sepal.Length, y = Petal.Width, color=Species, bg_colour=Species)
) +
geom_bgpoint(pointsize=4, gap_line_width = .1, bg_line_width = .3)
```

`geom_scattermore2` *geom_scattermore2*

Description

this add the background colour for the `geom_scattermore`

Usage

```
geom_scattermore2(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  interpolate = FALSE,
  pointsize = 0,
  pixels = c(512, 512),
  gap_colour = "white",
  gap_alpha = 1,
  bg_line_width = 0.3,
  gap_line_width = 0.1
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`stat` The statistical transformation to use on the data for this layer. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:

- A `Stat` ggproto subclass, for example `StatCount`.
- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`...` Other arguments passed on to `layer`.

`na.rm` If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `annotation_borders()`.

`interpolate` A logical value indicating whether to linearly interpolate the image (the alternative is to use nearest-neighbour interpolation, which gives a more blocky result). Default `FALSE`, passed to `rasterGrob`.

`pointsize` Radius of rasterized point. Use '0' for single pixels (fastest).

`pixels` Vector with X and Y resolution of the raster, default `c(512,512)`.

`gap_colour` colour of gap background between the bottom background and top point point layer, default is `white`.

gap_alpha	numeric the transparency of gap background colour, default is 1.
bg_line_width	numeric the line width of background point layer, default is 0.3.
gap_line_width	numeric the line width of gap between the background and top point point layer, default is .1.

Details

Aesthetics geom_scattermore2 understands the following aesthetics. Required aesthetics are displayed in bold and defaults are displayed for optional aesthetics:

- x.
- y.
- colour the colour of point, default is black.
- bg_colour the colour of background point, default is NA.
- alpha the transparency of colour, default is 1.
- subset subset the data frame which meet conditions to display.

Value

polygonal point layer

Author(s)

Shuangbin Xu

Examples

```
library(ggplot2)
ggplot(iris,
 aes(x= Sepal.Length, y = Petal.Width, color=Species, bg_colour=Species)
) +
geom_scattermore2(pointsize=4, gap_line_width = .1, bg_line_width = .3)
```

---

plot_lisa_feature *plot_lisa_feature*

Description

plot_lisa_feature

Usage

```
plot_lisa_feature(
  spe,
  lisa.res,
  features = NULL,
  assay.type = "logcounts",
  geom = geom_bgpoint,
  pointsize = 2,
  hlpointsize = 1.8,
  clustertype = "High",
  hlcolor = c("black"),
  gap_line_width = 0.1,
  bg_line_width = 0.3,
  facet_name = NULL,
  reduction = NULL,
  image.plot = FALSE,
  label_wrap_width = 30,
  ncol = 6,
  ...
)
```

Arguments

spe	SpatialExperiment or SingleCellExperiment object.
lisa.res	the result returned by SVP::runLISA().
features	selected features to be visualized, default is NULL.
assay.type	the assay name where data will be used from (e.g., 'data', 'counts'), default is 'logcounts'.
geom	the function of geometric layer, default is geom_bgpoint, other option is sc_geom_point.
pointsize	numeric the size of point, default is 2.
hlpointsize	numeric the size of point which contains corresbonding spatially variable gene(i.e., SVG), default is 1.8.
clustertype	cell type which is from the result of lisa.res, default is 'High'.
hlcolor	the color of circular line which enfolds the point that contains SVG, default is 'black'.
gap_line_width	numeric the line width of gap between the background and top point point layer, default is .1.
bg_line_width	numeric the line width of background point layer, default is 0.3.
facet_name	the name of facet used in facet_wrap(), default is NULL.
reduction	reduction method, default is NULL and will use the default setting store in the object
image.plot	logical whether display the image of spatial experiment, default is FALSE.
label_wrap_width	numeric maximum number of characters before wrapping the strip. default is 30.

ncol numeric the column number, default is 6.

... additional parameters pass to `scattermore::geom_scattermore()`

- `bg_colour` the colour of background point, default is `NA`. this character also can be set in `mappint`.
- `alpha` the transparency of colour, default is 1.

Value

ggplot object

Examples

```
## Not run:
library(ggplot2)
library(SingleCellExperiment) |> suppressPackageStartupMessages()
library(SpatialExperiment) |> suppressPackageStartupMessages()
library(STexampleData)
# create ExperimentHub instance
eh <- ExperimentHub()
# query STexampleData datasets
myfiles <- query(eh, "STexampleData")
ah_id <- myfiles$ah_id[myfiles$title == 'Visium_humanDLPFC']
spe <- myfiles[[ah_id]]
spe <- spe[, colData(spe)$in_tissue == 1]
spe <- suppressWarnings(scater::logNormCounts(spe))
genes <- c('MOBP', 'PCP4', 'SNAP25', 'HBB', 'IGKC', 'NPY')
target.features <- rownames(spe)[match(genes, rowData(spe)$gene_name)]
library(SVP)
lisa.res1 <- runLISA(spe,
                     assay.type='logcounts',
                     features=target.features[seq(2)],
                     weight.method='knn',
                     k=50)
plot_lisa_feature(spe, lisa.res=lisa.res1, features=target.features[seq(2)],
                  pointsize=2, hlpointsize=2, gap_line_width=.1)

## End(Not run)
```

`reexports` *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

ggfun `%<+%`

ggplot2 `aes`, `theme`

Value

Depending on the re-exported function

`scale_bg_color_manual` *Create your own discrete scale*

Description

Create your own discrete scale

Usage

```
scale_bg_colour_identity(
  name = waiver(),
  ...,
  guide = "none",
  aesthetics = "bg_colour"
)

scale_bg_colour_manual(
  ...,
  values,
  aesthetics = "bg_colour",
  breaks = waiver(),
  na.value = "grey50"
)
```

Arguments

`...` Arguments passed on to ggplot2::discrete_scale, ggplot2::discrete_scale

`scale_name` **[Deprecated]** The name of the scale that should be used for error messages associated with this scale.

`palette` A palette function that when called with a single integer argument (the number of levels in the scale) returns the values that they should take (e.g., `scales::pal_hue()`).

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- NULL for no labels
- waiver() for the default labels computed by the transformation object
- A character vector giving labels (must be same length as breaks)
- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

limits One of:

- NULL to use the default scale values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

na.translate Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify na.translate = FALSE.

drop Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE includes the levels in the factor. Please note that to display every level in a legend, the layer should use show.legend = TRUE.

fallback.palette Function to use when palette = NULL and the palette is not represented in the theme.

call The call used to construct the scale for reporting messages.

super The super class to use for the constructed scale

name The name of the scale. Used as the axis or legend title. If waiver(), the default, the name of the scale is taken from the first mapping used for that aesthetic. If NULL, the legend title will be omitted.

guide A function used to create a guide or its name. See guides() for more information.

aesthetics The names of the aesthetics that this scale works with.

values a set of aesthetic values to map data values to. If this is a named vector, then the values will be matched based on the names. If unnamed, values will be matched in order (usually alphabetical) with the limits of the scale. Any data values that don't match will be given na.value.

breaks One of:

- NULL for no breaks
- waiver() for the default breaks (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.

na.value If na.translate = TRUE, what aesthetic value should the missing values be displayed as? Does not apply to position scales where NA is always placed at the far right.

Value

bg_colour scale constructor

---

sc_dim *sc_dim*

Description

sc_dim

Usage

```
sc_dim(
  object,
  dims = c(1, 2),
  reduction = NULL,
  cells = NULL,
  slot = "data",
  mapping = NULL,
  geom = sc_geom_point,
  ...
)

## S4 method for signature 'Seurat'
sc_dim(
  object,
  dims = c(1, 2),
  reduction = NULL,
  cells = NULL,
  slot = "data",
  mapping = NULL,
  geom = sc_geom_point,
  ...
)

## S4 method for signature 'SingleCellExperiment'
sc_dim(
  object,
  dims = c(1, 2),
  reduction = NULL,
  cells = NULL,
  slot = "data",
  mapping = NULL,
  geom = sc_geom_point,
  ...
)
```

Arguments

object	Seurat object or SingleCellExperiment object
dims	selected dimensions (must be a two-length vector) that are used in visualization
reduction	reduction method, default is NULL and will use the default setting store in the object
cells	selected cells to plot (default is all cells)
slot	slot to pull expression data from (e.g., 'count' or 'data')
mapping	aesthetic mapping, the x and y is set internally, other character of geometric layer, such as color, size, alpha or (shape when geom = geom_point) can be set manually.
geom	the function of geometric layer, default is sc_geom_point, other geometric layer, such as geom_bgpoint or geom_point also works.
...	additional parameters pass to geom_scattermore2().

- bg_colour the colour of background point, default is NA. this character also can be set in mappint.
- gap_colour the colour of gap background, default is 'white'.
- bg_line_width the line width of background point, default is .3.
- gap_line_width the gap line width of background point, default is .1.
- alpha the transparency of colour, default is 1.
- subset subset the data frame which meet conditions to display. this should be set in mapping.

Value

dimension reduction plot

See Also

geom_scattermore;

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
clusters <- clusterCells(sce, assay.type = 'logcounts')
colLabels(sce) <- clusters
sce <- runUMAP(sce, assay.type = 'logcounts')
p1 <- sc_dim(sce, reduction = 'UMAP', mapping = aes(colour = Cell_Cycle))
p2 <- sc_dim(sce, reduction = 'UMAP')
f1 <- p1 + sc_dim_geom_label()
f2 <- p2 +
      sc_dim_geom_label(
        geom = shadowtext::geom_shadowtext,
```

```
  color='black',
  bg.color='white'
)
```

---

sc_dim_count *sc_dim_count*

Description

sc_dim_count

Usage

```
sc_dim_count(sc_dim_plot)
```

Arguments

sc_dim_plot	dimension reduction plot of single cell data

Value

a bar plot to present the cell numbers of different clusters

See Also

sc_dim()

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
clusters <- clusterCells(sce, assay.type = 'logcounts')
colLabels(sce) <- clusters
sce <- runUMAP(sce, assay.type = 'logcounts')
p <- sc_dim(sce, reduction = 'UMAP')
p1 <- sc_dim_count(p)
```

sc_dim_geom_ellipse *sc_dim_geom_ellipse*

Description

sc_dim_geom_ellipse

Usage

```
sc_dim_geom_ellipse(geom = stat_ellipse, mapping = NULL, level = 0.95, ...)
```

Arguments

geom	the layer function, default is stat_ellipse, other option is geom_mark_hull of ggforce.
mapping	aesthetic mapping
level	the level at which to draw an ellipse
...	additional parameters pass to the stat_ellipse

Value

layer of ellipse

See Also

stat_ellipse;

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
clusters <- clusterCells(sce, assay.type = 'logcounts')
colLabels(sce) <- clusters
sce <- runUMAP(sce, assay.type = 'logcounts')
p1 <- sc_dim(sce, reduction = 'UMAP', mapping = aes(colour = Cell_Cycle))
p2 <- sc_dim(sce, reduction = 'UMAP')
f1 <- p1 + sc_dim_geom_ellipse()
```

sc_dim_geom_feature *sc_dim_geom_feature*

Description

sc_dim_geom_feature

Usage

```
sc_dim_geom_feature(
  object,
  features,
  dims = c(1, 2),
  ncol = 3,
  ...,
  .fun = function(.data) dplyr::filter(.data, .data$value > 0)
)
```

Arguments

object	Seurat or SingleCellExperiment object
features	selected features (i.e., genes)
dims	selected dimensions (must be a two-length vector) that are used in visualization
ncol	number of facet columns if 'length(features) > 1'
...	additional parameters pass to 'scattermore::geom_scattermore()'
.fun	user defined function that will be applied to selected features (default is to filter out genes with no expression values)

Value

layer of points for selected features

See Also

sc_feature()

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
clusters <- clusterCells(sce, assay.type = 'logcounts')
colLabels(sce) <- clusters
sce <- runUMAP(sce, assay.type = 'logcounts')
```

```
p1 <- sc_dim(sce, reduction = 'UMAP')
set.seed(123)
genes <- rownames(sce) |> sample(6)
f1 <- p1 +
      sc_dim_geom_feature(
        object = sce,
        features = genes
      )
```

---

sc_dim_geom_label *sc_dim_geom_label*

Description

sc_dim_geom_label

Usage

```
sc_dim_geom_label(geom = ggplot2::geom_text, mapping = NULL, ...)
```

Arguments

geom	geometric layer (default: geom_text) to display the lables
mapping	aesthetic mapping
...	additional parameters pass to the geom

Value

layer of labels

See Also

sc_dim_geom_label()

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
clusters <- clusterCells(sce, assay.type = 'logcounts')
colLabels(sce) <- clusters
sce <- runUMAP(sce, assay.type = 'logcounts')
p1 <- sc_dim(sce, reduction = 'UMAP', mapping = aes(colour = Cell_Cycle))
p2 <- sc_dim(sce, reduction = 'UMAP')
f1 <- p1 + sc_dim_geom_label()
```

sc_dim_geom_sub *sc_dim_geom_subset*

Description

sc_dim_geom_subset

Usage

```
sc_dim_geom_sub(mapping = NULL, subset, .column = "ident", ...)
```

Arguments

mapping	aesthetic mapping
subset	subset of clusters to be displayed
.column	which column represents cluster (e.g., 'ident')
...	additional parameters pass to sc_geom_point

Value

plot with a layer of specified clusters

See Also

sc_dim_geom_sub

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
clusters <- clusterCells(sce, assay.type = 'logcounts')
colLabels(sce) <- clusters
sce <- runUMAP(sce, assay.type = 'logcounts')
p1 <- sc_dim(sce, reduction = 'UMAP')
f1 <- p1 + sc_dim_geom_sub(subset = c(1, 2), .column = 'label', bg_colour='black')
```

sc_dim_sub *sc_dim_sub*

Description

sc_dim_sub

Usage

```
sc_dim_sub(subset, .column = "ident")
```

Arguments

subset	subset of clusters to be displayed
.column	which column represents cluster (e.g., 'ident')

Value

update plot with only subset displayed

See Also

sc_dim

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
clusters <- clusterCells(sce, assay.type = 'logcounts')
colLabels(sce) <- clusters
sce <- runUMAP(sce, assay.type = 'logcounts')
p1 <- sc_dim(sce, reduction = 'UMAP')
f1 <- p1 + sc_dim_sub(subset = c(1, 2), .column = 'label')
```

sc_dot *sc_dot*

Description

sc_dot

Usage

```
sc_dot(
  object,
  features,
  group.by = NULL,
  split.by = NULL,
  cols = c("lightgrey", "blue"),
  col.min = -2.5,
  col.max = 2.5,
  dot.min = 0,
  dot.scale = 6,
  slot = "data",
  .fun = NULL,
  mapping = NULL,
  scale = TRUE,
  scale.by = "radius",
  scale.min = NA,
  scale.max = NA,
  cluster.idents = FALSE,
  ...
)

## S4 method for signature 'Seurat'
sc_dot(
  object,
  features,
  group.by = NULL,
  split.by = NULL,
  cols = c("lightgrey", "blue"),
  col.min = -2.5,
  col.max = 2.5,
  dot.min = 0,
  dot.scale = 6,
  slot = "data",
  .fun = NULL,
  mapping = NULL,
  scale = TRUE,
  scale.by = "radius",
  scale.min = NA,
```

```
  scale.max = NA,
  cluster.idents = FALSE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
sc_dot(
  object,
  features,
  group.by = NULL,
  split.by = NULL,
  cols = c("lightgrey", "blue"),
  col.min = -2.5,
  col.max = 2.5,
  dot.min = 0,
  dot.scale = 6,
  slot = "data",
  .fun = NULL,
  mapping = NULL,
  scale = TRUE,
  scale.by = "radius",
  scale.min = NA,
  scale.max = NA,
  cluster.idents = FALSE,
  ...
)
```

Arguments

`object`	Seurat or SingleCellExperiment object
`features`	selected features
`group.by`	grouping factor
`split.by`	additional split factor
`cols`	colors of the points
`col.min`	minimum scaled averaged expression threshold
`col.max`	maximum scaled averaged expression threshold
`dot.min`	the threshold of percentage of cells for the the smallest dot
`dot.scale`	Scaling factor for size of points
`slot`	slot to pull expression data from (e.g., 'count' or 'data')
`.fun`	user defined function that will be applied to selected features (default is NULL and there is no data operation)
`mapping`	aesthetic mapping
`scale`	whether to scale the expression value (default to TRUE)
`scale.by`	scale the size of the points by `size` or `radius`
`scale.min`	lower limit of scaling

scale.max	upper limit of scaling
cluster.idents	Order identities by hierarchical clusters based on average expression and perventage of expression (default is FALSE)
...	additional parameters pass to 'ggplot2::geom_point()'

Value

dot plot to visualize feature expression distribution

See Also

DotPlot;

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
set.seed(123)
genes <- rownames(sce) |> sample(6)
sc_dot(sce, genes[1:5], 'Treatment', slot = 'logcounts')
```

---

sc_feature *sc_feature*

Description

sc_feature

Usage

```
sc_feature(
  object,
  features,
  dims = c(1, 2),
  reduction = NULL,
  cells = NULL,
  slot = "data",
  mapping = NULL,
  ncol = 3,
  density = FALSE,
  grid.n = 100,
  joint = FALSE,
  joint.fun = prod,
```

```
  common.legend = TRUE,
  geom = sc_geom_point,
  ...
)

## S4 method for signature 'Seurat'
sc_feature(
  object,
  features,
  dims = c(1, 2),
  reduction = NULL,
  cells = NULL,
  slot = "data",
  mapping = NULL,
  ncol = 3,
  density = FALSE,
  grid.n = 100,
  joint = FALSE,
  joint.fun = prod,
  common.legend = TRUE,
  geom = sc_geom_point,
  ...
)

## S4 method for signature 'SingleCellExperiment'
sc_feature(
  object,
  features,
  dims = c(1, 2),
  reduction = NULL,
  cells = NULL,
  slot = "data",
  mapping = NULL,
  ncol = 3,
  density = FALSE,
  grid.n = 100,
  joint = FALSE,
  joint.fun = prod,
  common.legend = TRUE,
  geom = sc_geom_point,
  ...
)
```

Arguments

object	Seurat object
features	selected features (i.e., genes)
dims	selected dimensions (must be a two-length vector) that are used in visualization

reduction	reduction method, default is NULL and will use the default setting store in the object
cells	selected cells to plot (default is all cells)
slot	slot to pull expression data from (e.g., 'count' or 'data')
mapping	aesthetic mapping
ncol	number of facet columns if 'length(features) > 1'
density	whether plot the 2D weighted kernel density, default is FALSE.
grid.n	number of grid points in the two directions to estimate 2D weighted kernel density, default is 100.
joint	whether joint the multiple features with joint.fun, default is FALSE.
joint.fun	how to joint the multiple features if joint=TRUE, default is prod.
common.legend	whether to use facet_wrap to display the multiple features, default is TRUE.
geom	the function of geometric layer, default is sc_geom_point, other geometric layer, such as geom_bgpoint or geom_point also works.
...	additional parameters pass to 'scattermore::geom_scattermore()'

- bg_colour the colour of background point, default is NA. this character also can be set in mappint.
- gap_colour the colour of gap background, default is 'white'.
- bg_line_width the line width of background point, default is .3.
- gap_line_width the gap line width of background point, default is .1.
- alpha the transparency of colour, default is 1.
- subset subset the data frame which meet conditions to display. this should be set in mapping.

Value

dimension reduction plot colored by selected features

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
clusters <- clusterCells(sce, assay.type = 'logcounts')
colLabels(sce) <- clusters
sce <- runTSNE(sce, assay.type = 'logcounts')
set.seed(123)
genes <- rownames(sce) |> sample(6)
p1 <- sc_feature(sce, genes[1], slot='logcounts', reduction = 'TSNE')
p2 <- sc_feature(sce, genes, slot='logcounts', reduction = 'TSNE')
f1 <- sc_dim(sce, slot='logcounts', reduction = 'TSNE') +
      sc_dim_geom_feature(sce, genes[1], color='black')
f2 <- sc_dim(sce, alpha=.3, slot='logcounts', reduction = 'TSNE') +
```

```
    ggnewscale::new_scale_color() +
    sc_dim_geom_feature(sce, genes, mapping=aes(color=features)) +
    scale_color_viridis_d()
p1 + p2 + f1 + f2
# The features can also be specified the variables from
# colData or reducedDims
pp <- sc_feature(sce, features = 'sizeFactor', reduction='TSNE', geom=geom_bgpoint)
pp
```

`sc_geom_annot` *add the annotation layer for ggsc object*

Description

add the annotation layer for ggsc object

Usage

```
sc_geom_annot(
  data = NULL,
  mapping = NULL,
  pointsize = 2,
  pixels = c(512, 512),
  gap_colour = "white",
  gap_alpha = 1,
  bg_line_width = 0.3,
  gap_line_width = 0.1,
  show.legend = NA,
  na.rm = FALSE,
  ...
)
```

Arguments

`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, will override the plot data. the `data.frame` should have a barcode id or features column. A `function` will be called with a single argument, the plot data. The return value must be a 'data.frame', and will be used as the layer data. A `function` can be created from a 'formula' (e.g. '~ head(.x, 10)').
`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`pointsize`	Radius of rasterized point. Use '0' for single pixels (fastest).
`pixels`	Vector with X and Y resolution of the raster, default `c(512,512)`.
`gap_colour`	colour of gap background between the bottom background and top point point layer, default is `white`.

gap_alpha	numeric the transparency of gap background colour, default is 1.
bg_line_width	numeric the line width of background point layer, default is 0.3.
gap_line_width	numeric the line width of gap between the background and top point point layer, default is .1.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
...	Other arguments passed on to layer.

Value

layer object

---

sc_geom_point *sc_geom_point*

Description

sc_geom_point

Usage

```
sc_geom_point(mapping = NULL, ...)
```

Arguments

mapping	aesthetic mapping
...	additional parameters pass to 'scattermore::geom_scattermore()'

Value

layer of points

See Also

sc_dim() and sc_feature()

Examples

```
library(ggplot2)
ggplot(iris,
 aes(x= Sepal.Length, y = Petal.Width, color=Species)
) +
sc_geom_point()
```

sc_spatial *sc_spatial*

Description

sc_spatial

Usage

```
sc_spatial(
  object,
  features = NULL,
  sample.id = NULL,
  image.id = NULL,
  slot = "data",
  plot.pie = FALSE,
  pie.radius.scale = 0.3,
  image.plot = TRUE,
  image.first.operation = "rotate",
  image.rotate.degree = NULL,
  image.mirror.axis = NULL,
  remove.point = FALSE,
  mapping = NULL,
  ncol = 6,
  density = FALSE,
  grid.n = 100,
  joint = FALSE,
  joint.fun = prod,
  common.legend = TRUE,
  pointsize = 5,
  geom = sc_geom_point,
  reduction = NULL,
  dims = NULL,
  ...
)

## S4 method for signature 'Seurat'
sc_spatial(
  object,
  features = NULL,
  sample.id = NULL,
  image.id = NULL,
  slot = "data",
  plot.pie = FALSE,
  pie.radius.scale = 0.3,
  image.plot = TRUE,
  image.first.operation = "rotate",
```

```
  image.rotate.degree = NULL,
  image.mirror.axis = NULL,
  remove.point = FALSE,
  mapping = NULL,
  ncol = 6,
  density = FALSE,
  grid.n = 100,
  joint = FALSE,
  joint.fun = prod,
  common.legend = TRUE,
  pointsize = 5,
  geom = sc_geom_point,
  reduction = NULL,
  dims = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment'
sc_spatial(
  object,
  features = NULL,
  sample.id = NULL,
  image.id = NULL,
  slot = 1,
  plot.pie = FALSE,
  pie.radius.scale = 0.3,
  image.plot = TRUE,
  image.first.operation = "rotate",
  image.rotate.degree = NULL,
  image.mirror.axis = "v",
  remove.point = FALSE,
  mapping = NULL,
  ncol = 6,
  density = FALSE,
  grid.n = 100,
  joint = FALSE,
  joint.fun = prod,
  common.legend = TRUE,
  pointsize = 5,
  geom = sc_geom_point,
  reduction = NULL,
  dims = NULL,
  ...
)
```

Arguments

object	Seurat object

`features` selected features to be visualized

`sample.id` the index name of sample id, which only work with SingleCellExperiment or SpatialExperiment.

`image.id` the index name of image id, which only work with SingleCellExperiment or SpatialExperiment.

`slot` if plotting a feature, which data will be used (e.g., 'data', 'counts'), the assay name if object is SingleCellExperiment or SpatialExperiment.

`plot.pie` logical whether plot the features with pie, default is `FALSE`.

`pie.radius.scale` numeric scale to the radius of pie only work with `plot.pie=TRUE`, default is 0.3.

`image.plot` whether to display the issue image as background.

`image.first.operation` character which the first operation to image, 'rotate' or 'mirror', default is 'rotate'.

`image.rotate.degree` integer the degree to ratate image, default is NULL.

`image.mirror.axis` character the direction to mirror the image, default is 'h'.

`remove.point` whether to remove the spot points, it is nice if your just view the issue image, default is FALSE.

`mapping` aesthetic mapping, default is NULL.

`ncol` integer number of facet columns if 'length(features) > 1', default is 6.

`density` whether plot the 2D weighted kernel density, default is FALSE.

`grid.n` number of grid points in the two directions to estimate 2D weighted kernel density, default is 100.

`joint` whether joint the multiple features with `joint.fun`, default is FALSE.

`joint.fun` how to joint the multiple features if `joint = TRUE`, default is prod.

`common.legend` whether to use `facet_wrap` to display the multiple `features`, default is TRUE.

`pointsize` the size of point, default is 5.

`geom` the layer of point, default is `sc_geom_point`, other option is `geom_bgpoint`.

`reduction` reduction method name, default is NULL and will use the spatial coordinates.

`dims` selected dimensions (must be a two-length vector) that are used in visualization, which works with `reduction` argument.

`...` additional parameters, see also `geom_scattermore2()`.

- `bg_colour` the colour of background point, default is `NA`. this character also can be set in `mappint`.
- `gap_colour` the colour of gap background, default is `'white'`.
- `bg_line_width` the line width of background point, default is `.3`.
- `gap_line_width` the gap line width of background point, default is `.1`.
- `alpha` the transparency of colour, default is 1.
- `subset` subset the data frame which meet conditions to display. this should be set in `mapping`.

Value

ggplot object

Examples

```
## Not run:
library(STexampleData)
# create ExperimentHub instance
eh <- ExperimentHub()
# query STexampleData datasets
myfiles <- query(eh, "STexampleData")
ah_id <- myfiles$ah_id[myfiles$title == 'Visium_humanDLPFC']
spe <- myfiles[[ah_id]]
spe <- spe[, colData(spe)$in_tissue == 1]
set.seed(123)
genes <- rownames(spe) |> sample(6)
p <- sc_spatial(spe, features = genes,
                image.rotate.degree = -90,
                image.mirror.axis = NULL,
                ncol = 3)
# The features also can be specified
# the variables from colData or reducedDims.
p1 <- sc_spatial(spe, features = 'cell_count',
                 image.rotate.degree = -90, image.mirror.axis = NULL)

## End(Not run)
```

---

sc_violin *sc_violin*

Description

sc_violin

Usage

```
sc_violin(
  object,
  features,
  cells = NULL,
  slot = "data",
  .fun = NULL,
  mapping = NULL,
  ncol = 3,
  geom = geom_violin,
  ...
)
```

```
## S4 method for signature 'Seurat'
sc_violin(
  object,
  features,
  cells = NULL,
  slot = "data",
  .fun = NULL,
  mapping = NULL,
  ncol = 3,
  geom = geom_violin,
  ...
)

## S4 method for signature 'SingleCellExperiment'
sc_violin(
  object,
  features,
  cells = NULL,
  slot = "data",
  .fun = NULL,
  mapping = NULL,
  ncol = 3,
  geom = geom_violin,
  ...
)
```

Arguments

object	Seurat object
features	selected features
cells	selected cells to plot (default is all cells)
slot	slot to pull expression data from (e.g., 'count' or 'data')
.fun	user defined function that will be applied to selected features (default is NULL and there is no data operation)
mapping	aesthetic mapping
ncol	number of facet columns if 'length(features) > 1'
geom	the geom function, default is geom_violin, other option is geom_boxplot
...	additional parameters pass to 'ggplot2::geom_geom_violin()'

Value

violin plot to visualize feature expression distribution

See Also

geom_violin;

Examples

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- suppressWarnings(logNormCounts(sce))
clusters <- clusterCells(sce, assay.type = 'logcounts')
colLabels(sce) <- clusters
sce <- runUMAP(sce, assay.type = 'logcounts')
set.seed(123)
genes <- rownames(sce) |> sample(6)
sc_violin(sce, genes[1], slot = 'logcounts')
sc_violin(sce, genes[1], slot = 'logcounts',
     .fun=function(d) dplyr::filter(d, value > 0)
     ) +
     ggforce::geom_sina(size=.1)
sc_violin(sce, genes, slot = 'logcounts') +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```

Index

∗ **internal**
    ggsc-package, 3
    reexports, 11
%<+% *(*reexports*)*, 11
%<+%, *11*

aes, *11*
aes *(*reexports*)*, 11
aes(), *5*, *7*
annotation_borders(), *6*, *8*

CalWkdeCpp, 3

DotPlot, *24*
draw_key_bgpoint, 4

fortify(), *5*, *8*

geom_bgpoint, 5
geom_point, *5*
geom_scattermore, *7*, *15*
geom_scattermore2, 7
geom_violin, *33*
ggplot(), *5*, *7*
ggplot2::discrete_scale, *12*
ggsc *(*ggsc-package*)*, 3
ggsc-package, 3
guides(), *13*

lambda, *12, 13*
layer, *6*, *8*, *28*
layer position, *6*, *8*
layer stat, *6*, *8*

plot_lisa_feature, 9

rasterGrob, *8*
reexports, 11

sc_dim, 14, *21*
sc_dim(), *16*, *28*
sc_dim,Seurat *(*sc_dim*)*, 14
sc_dim,Seurat-method *(*sc_dim*)*, 14
sc_dim,SingleCellExperiment *(*sc_dim*)*, 14
sc_dim,SingleCellExperiment-method *(*sc_dim*)*, 14
sc_dim_count, 16
sc_dim_geom_ellipse, 17
sc_dim_geom_feature, 18
sc_dim_geom_label, 19
sc_dim_geom_label(), *19*
sc_dim_geom_sub, *20*, 20
sc_dim_sub, 21
sc_dot, 22
sc_dot,Seurat *(*sc_dot*)*, 22
sc_dot,Seurat-method *(*sc_dot*)*, 22
sc_dot,SingleCellExperiment *(*sc_dot*)*, 22
sc_dot,SingleCellExperiment-method *(*sc_dot*)*, 22
sc_feature, 24
sc_feature(), *18*, *28*
sc_feature,Seurat *(*sc_feature*)*, 24
sc_feature,Seurat-method *(*sc_feature*)*, 24
sc_feature,SingleCellExperiment *(*sc_feature*)*, 24
sc_feature,SingleCellExperiment-method *(*sc_feature*)*, 24
sc_geom_annot, 27
sc_geom_point, 28
sc_spatial, 29
sc_spatial,Seurat *(*sc_spatial*)*, 29
sc_spatial,Seurat-method *(*sc_spatial*)*, 29
sc_spatial,SingleCellExperiment *(*sc_spatial*)*, 29
sc_spatial,SingleCellExperiment-method *(*sc_spatial*)*, 29
sc_violin, 32
sc_violin,Seurat *(*sc_violin*)*, 32

sc_violin,Seurat-method *(sc_violin)*, 32
sc_violin,SingleCellExperiment
(sc_violin), 32
sc_violin,SingleCellExperiment-method
(sc_violin), 32
scale_bg_color_identity
(scale_bg_color_manual), 12
scale_bg_color_manual, 12
scale_bg_colour_discrete
(scale_bg_color_manual), 12
scale_bg_colour_identity
(scale_bg_color_manual), 12
scale_bg_colour_manual
(scale_bg_color_manual), 12
scales::pal_hue(), *12*
stat_ellipse, *17*

theme, *11*
theme *(reexports)*, 11