

# Package ‘ggsc’

November 29, 2024

**Title** Visualizing Single Cell and Spatial Transcriptomics

**Version** 1.4.0

**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'.

**Imports** Rcpp, RcppParallel, cli, dplyr, ggfun (>= 0.1.5), ggplot2, grDevices, grid, methods, rlang, scattermore, stats, Seurat, SingleCellExperiment, SummarizedExperiment, tidydr, tidy, 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** 7.3.2

**Config/testthat/edition** 3

**LinkingTo** Rcpp, RcppArmadillo, RcppParallel

**Remotes** YuLab-SMU/SVP

**git\_url** <https://git.bioconductor.org/packages/ggsc>

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** 2ab4b64

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.20

**Date/Publication** 2024-11-28

**Author** Guangchuang Yu [aut, cre, cph]

(<https://orcid.org/0000-0002-6485-8781>),

Shuangbin Xu [aut] (<https://orcid.org/0000-0003-3513-5362>),

Noriaki Sato [ctb]

**Maintainer** Guangchuang Yu <guangchuangyu@gmail.com>

## Contents

ggsc-package	2
CalWkdeC++	3
draw_key_bgpoint	3
geom_bgpoint	4
geom_scattermore2	6
plot_lisa_feature	9
reexports	10
scale_bg_color_manual	11
sc_dim	12
sc_dim_count	14
sc_dim_geom_ellipse	15
sc_dim_geom_feature	16
sc_dim_geom_label	17
sc_dim_geom_sub	18
sc_dim_sub	18
sc_dot	19
sc_feature	22
sc_geom_annot	24
sc_geom_point	25
sc_spatial	26
sc_violin	29

**Index** 31

---

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://github.com/YuLab-SMU/ggsc(devel))
- [https://yulab-smu.top/ggsc/\(docs\)](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	<i>geom_bgpoint</i>
--------------	---------------------

---

**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 <a href="#">aes()</a> . If specified and <code>inherit.aes = TRUE</code> (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 <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <a href="#">fortify()</a> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code> ).
stat	The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: <ul style="list-style-type: none"> <li>• A <code>Stat</code> ggproto subclass, for example <code>StatCount</code>.</li> <li>• A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>.</li> </ul>

- 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: <ul style="list-style-type: none"> <li>• The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position.</li> <li>• A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter".</li> <li>• For more information and other ways to specify the position, see the <a href="#">layer position</a> documentation.</li> </ul>
...	Other arguments passed on to <a href="#">layer</a> .
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.
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. <a href="#">borders()</a> .
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 <code>geom_bgpoint</code>

### Details

- `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.

### Aesthetics

`geom_bgpoint()` understands the following aesthetics (required aesthetics are in bold):

- `x`
- `y`
- `alpha`
- `bg_colour`
- **`colour`**
- **`fill`**

- [group](#)
- [shape](#)
- [size](#)
- [stroke](#)
- [subset](#)

Learn more about setting these aesthetics in `vignette("ggplot2-specs")`.

### 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 <code>aes()</code> . If specified and <code>inherit.aes = TRUE</code> (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 <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code> ).
stat	The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following: <ul style="list-style-type: none"> <li>• A <code>Stat</code> ggproto subclass, for example <code>StatCount</code>.</li> <li>• A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as "count".</li> <li>• For more information and other ways to specify the stat, see the <a href="#">layer stat</a> documentation.</li> </ul>
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 <code>position</code> argument accepts the following: <ul style="list-style-type: none"> <li>• The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position.</li> <li>• A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter".</li> <li>• For more information and other ways to specify the position, see the <a href="#">layer position</a> documentation.</li> </ul>
...	Other arguments passed on to <a href="#">layer</a> .
na.rm	If <code>FALSE</code> , the default, missing values are removed with a warning. If <code>TRUE</code> , missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If <code>FALSE</code> , 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. <code>borders()</code> .
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 <code>FALSE</code> , passed to <code>rasterGrob</code> .
pointsize	Radius of rasterized point. Use '0' for single pixels (fastest).
pixels	Vector with X and Y resolution of the raster, default <code>c(512, 512)</code> .

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

- 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

### Aesthetics

geom\_scattermore2() understands the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- **alpha**
- **bg\_colour**
- **colour**
- **fill**
- **group**
- **shape**
- **size**
- **stroke**
- **subset**

Learn more about setting these aesthetics in vignette("ggplot2-specs").

### 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,
  ...
)
```

## Arguments

spe	SpatialExperiment or SingleCellExperiment object.
lisa.res	the result returned by <code>SVP::runLISA()</code> .
features	selected features to be visualized, default is <code>NULL</code> .
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 <code>geom_bgpoint</code> , other option is <code>sc_geom_point</code> .
pointsize	numeric the size of point, default is 2.
hlpointsize	numeric the size of point which contains corresponding spatially variable gene(i.e., SVG), default is 1.8.
clustertype	cell type which is from the result of <code>lisa.res</code> , 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 <code>facet_wrap()</code> , default is <code>NULL</code> .
reduction	reduction method, default is <code>NULL</code> 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.  
`...` 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)$in_tissue == 1]
spe <- scatter::logNormCounts(spe)
genes <- c('MOBP', 'PCP4', 'SNAP25', 'HBB', 'IGKC', 'NPY')
target.features <- rownames(spe)[match(genes, rowData(spe)$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()`).

`labels` One of:

- 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 `?plot-math` 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`.

	<p><code>drop</code> Should unused factor levels be omitted from the scale? The default, <code>TRUE</code>, uses the levels that appear in the data; <code>FALSE</code> includes the levels in the factor. Please note that to display every level in a legend, the layer should use <code>show.legend = TRUE</code>.</p> <p><code>call</code> The call used to construct the scale for reporting messages.</p> <p><code>super</code> The super class to use for the constructed scale</p>
<code>name</code>	The name of the scale. Used as the axis or legend title. If <code>waiver()</code> , the default, the name of the scale is taken from the first mapping used for that aesthetic. If <code>NULL</code> , the legend title will be omitted.
<code>guide</code>	A function used to create a guide or its name. See <code>guides()</code> for more information.
<code>aesthetics</code>	The names of the aesthetics that this scale works with.
<code>values</code>	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 <code>na.value</code> .
<code>breaks</code>	One of: <ul style="list-style-type: none"> <li>• <code>NULL</code> for no breaks</li> <li>• <code>waiver()</code> for the default breaks (the scale limits)</li> <li>• A character vector of breaks</li> <li>• A function that takes the limits as input and returns breaks as output. Also accepts rlang <code>lambda</code> function notation.</li> </ul>
<code>na.value</code>	If <code>na.translate = TRUE</code> , 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

---

<code>sc_dim</code>	<i>sc_dim</i>
---------------------	---------------

---

## 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(). <ul style="list-style-type: none"> <li>• bg_colour the colour of background point, default is NA. this character also can be set in mappint.</li> <li>• gap_colour the colour of gap background, default is 'white'.</li> <li>• bg_line_width the line width of background point, default is .3.</li> <li>• gap_line_width the gap line width of background point, default is .1.</li> <li>• alpha the transparency of colour, default is 1.</li> <li>• subset subset the data frame which meet conditions to display. this should be set in mapping.</li> </ul>

**Value**

dimension reduction plot

**See Also**

[geom\\_scattermore](#);

**Examples**

```
library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- 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 <- 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 <- 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 <- 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 labels
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 <- 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 <- 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 <- 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.ident	Order identities by hierarchical clusters based on average expression and percentage 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 <- 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()' <ul style="list-style-type: none"> <li>• bg_colour the colour of background point, default is NA. this character also can be set in mappint.</li> <li>• gap_colour the colour of gap background, default is 'white'.</li> <li>• bg_line_width the line width of background point, default is .3.</li> <li>• gap_line_width the gap line width of background point, default is .1.</li> <li>• alpha the transparency of colour, default is 1.</li> <li>• subset subset the data frame which meet conditions to display. this should be set in mapping.</li> </ul>

### Value

dimension reduction plot colored by selected features

**Examples**

```

library(scuttle)
library(scater)
library(scran)
library(ggplot2)
sce <- mockSCE()
sce <- 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 <code>data.frame</code> , will override the plot data. the <code>data.frame</code> 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 <code>aes()</code> . If specified and <code>inherit.aes = TRUE</code> (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 <code>c(512, 512)</code> .
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.
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 <a href="#">layer</a> .

**Value**

layer object

---

sc_geom_point	<i>sc_geom_point</i>
---------------	----------------------

---

**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,
  ...
)
```

## 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,
    ...
)

## 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,
  ...
)

```

### 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.

<code>plot.pie</code>	logical whether plot the features with pie, default is FALSE.
<code>pie.radius.scale</code>	numeric scale to the radius of pie only work with <code>plot.pie=TRUE</code> , default is 0.3.
<code>image.plot</code>	whether to display the issue image as background.
<code>image.first.operation</code>	character which the first operation to image, 'rotate' or 'mirror', default is 'rotate'.
<code>image.rotate.degree</code>	integer the degree to rotate image, default is NULL.
<code>image.mirror.axis</code>	character the direction to mirror the image, default is 'h'.
<code>remove.point</code>	whether to remove the spot points, it is nice if your just view the issue image, default is FALSE.
<code>mapping</code>	aesthetic mapping, default is NULL.
<code>ncol</code>	integer number of facet columns if 'length(features) > 1', default is 6.
<code>density</code>	whether plot the 2D weighted kernel density, default is FALSE.
<code>grid.n</code>	number of grid points in the two directions to estimate 2D weighted kernel density, default is 100.
<code>joint</code>	whether joint the multiple features with <code>joint.fun</code> , default is FALSE.
<code>joint.fun</code>	how to joint the multiple features if <code>joint = TRUE</code> , default is <code>prod</code> .
<code>common.legend</code>	whether to use <code>facet_wrap</code> to display the multiple features, default is TRUE.
<code>pointsize</code>	the size of point, default is 5.
<code>geom</code>	the layer of point, default is <code>sc_geom_point</code> , other option is <code>geom_bgpoint</code> .
<code>...</code>	additional parameters, see also <code>geom_scattermore2()</code> . <ul style="list-style-type: none"> <li>• <code>bg_colour</code> the colour of background point, default is NA. this character also can be set in <code>mapping</code>.</li> <li>• <code>gap_colour</code> the colour of gap background, default is 'white'.</li> <li>• <code>bg_line_width</code> the line width of background point, default is .3.</li> <li>• <code>gap_line_width</code> the gap line width of background point, default is .1.</li> <li>• <code>alpha</code> the transparency of colour, default is 1.</li> <li>• <code>subset</code> subset the data frame which meet conditions to display. this should be set in <code>mapping</code>.</li> </ul>

## 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 <- 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, 2

reexports, 10

%<+% (reexports), 10

%<+%, 10

aes, 10

aes (reexports), 10

aes(), 4, 7

alpha, 5, 8

borders(), 5, 7

CalWkdeCpp, 3

colour, 5, 8

DotPlot, 21

draw\_key\_bgpoint, 3

fill, 5, 8

fortify(), 4, 7

geom\_bgpoint, 4

geom\_point, 4

geom\_scattermore, 6, 14

geom\_scattermore2, 6

geom\_violin, 30

ggplot(), 4, 7

ggplot2::discrete\_scale, 11

ggsc (ggsc-package), 2

ggsc-package, 2

group, 6, 8

guides(), 12

lambda, 11, 12

layer, 5, 7, 25

layer position, 5, 7

layer stat, 5, 7

plot\_lisa\_feature, 9

rasterGrob, 7

reexports, 10

sc\_dim, 12, 19

sc\_dim(), 14, 25

sc\_dim, Seurat (sc\_dim), 12

sc\_dim, Seurat-method (sc\_dim), 12

sc\_dim, SingleCellExperiment (sc\_dim), 12

sc\_dim, SingleCellExperiment-method  
(sc\_dim), 12

sc\_dim\_count, 14

sc\_dim\_geom\_ellipse, 15

sc\_dim\_geom\_feature, 16

sc\_dim\_geom\_label, 17

sc\_dim\_geom\_label(), 17

sc\_dim\_geom\_sub, 18, 18

sc\_dim\_sub, 18

sc\_dot, 19

sc\_dot, Seurat (sc\_dot), 19

sc\_dot, Seurat-method (sc\_dot), 19

sc\_dot, SingleCellExperiment (sc\_dot), 19

sc\_dot, SingleCellExperiment-method  
(sc\_dot), 19

sc\_feature, 22

sc\_feature(), 16, 25

sc\_feature, Seurat (sc\_feature), 22

sc\_feature, Seurat-method (sc\_feature),  
22

sc\_feature, SingleCellExperiment  
(sc\_feature), 22

sc\_feature, SingleCellExperiment-method  
(sc\_feature), 22

sc\_geom\_annot, 24

sc\_geom\_point, 25

sc\_spatial, 26

sc\_spatial, Seurat (sc\_spatial), 26

sc\_spatial, Seurat-method (sc\_spatial),  
26

sc\_spatial, SingleCellExperiment  
(sc\_spatial), 26

sc\_spatial, SingleCellExperiment-method  
(sc\_spatial), 26

sc\_violin, 29

sc\_violin, Seurat (sc\_violin), 29

sc\_violin, Seurat-method (sc\_violin), 29

sc\_violin, SingleCellExperiment  
(sc\_violin), 29

sc\_violin, SingleCellExperiment-method  
    (sc\_violin), 29

scale\_bg\_color\_identity  
    (scale\_bg\_color\_manual), 11

scale\_bg\_color\_manual, 11

scale\_bg\_colour\_discrete  
    (scale\_bg\_color\_manual), 11

scale\_bg\_colour\_identity  
    (scale\_bg\_color\_manual), 11

scale\_bg\_colour\_manual  
    (scale\_bg\_color\_manual), 11

scales::pal\_hue(), 11

shape, 6, 8

size, 6, 8

stat\_ellipse, 15

theme, 10

theme (reexports), 10

x, 5, 8

y, 5, 8