

# Package ‘ggsc’

October 14, 2024

**Title** Visualizing Single Cell and Spatial Transcriptomics

**Version** 1.2.0

**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, ggplot2, grDevices, grid, methods, rlang, scattermore, stats, Seurat, SingleCellExperiment, SummarizedExperiment, tidydr, tidyr, tibble, utils, RColorBrewer, yulab.utils, scales

**Suggests** aplot, BiocParallel, forcats, ggforce, ggnewscale, igrph, knitr, ks, Matrix, prettydoc, rmarkdown, scan, scater, scatterpie, scuttle, shadowtext, sf, SeuratObject, SpatialExperiment, SExampleData, testthat (>= 3.0.0)

**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** 7.3.1

**Config/testthat/edition** 3

**LinkingTo** Rcpp, RcppArmadillo, RcppParallel

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

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** fd0d672

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-10-13

**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
CalWkdeCpp . . . . .	3
draw_key_scattermore2 . . . . .	4
geom_scattermore2 . . . . .	4
reexports . . . . .	7
scale_bg_color_manual . . . . .	7
sc_dim . . . . .	9
sc_dim_count . . . . .	11
sc_dim_geom_ellipse . . . . .	11
sc_dim_geom_feature . . . . .	12
sc_dim_geom_label . . . . .	13
sc_dim_geom_sub . . . . .	14
sc_dim_sub . . . . .	15
sc_dot . . . . .	16
sc_feature . . . . .	19
sc_geom_point . . . . .	21
sc_spatial . . . . .	22
sc_violin . . . . .	25
<b>Index</b>	<b>28</b>

---

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>
- 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 <code>bandwidth.nrd</code> ), A scalar value will be taken to apply to both directions (see <code>ks::hpi</code> ).
adjust	numeric value to adjust to bandwidth, default is 1.
n	number of grid points in the two directions, default is 400.

---

draw\_key\_scattermore2 *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_scattermore2(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\_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 <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, either as a <code>ggproto</code> <code>Geom</code> subclass or as a string naming the stat stripped of the <code>stat_</code> prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use <code>position_jitter</code> ), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	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. <a href="#">borders()</a> .
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 <a href="#">rasterGrob</a> .
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)
```

---

reexports	<i>Objects exported from other packages</i>
-----------	---

---

**Description**

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

**ggplot2** [aes](#), [theme](#)

**Value**

Depending on the re-exported function

---

scale_bg_color_manual	<i>Create your own discrete scale</i>
-----------------------	---------------------------------------

---

**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\(\)](#)).

	<p>labels One of:</p> <ul style="list-style-type: none"> <li>• NULL for no labels</li> <li>• <code>waiver()</code> for the default labels computed by the transformation object</li> <li>• A character vector giving labels (must be same length as breaks)</li> <li>• An expression vector (must be the same length as breaks). See <code>?plot-math</code> for details.</li> <li>• A function that takes the breaks as input and returns labels as output. Also accepts rlang <code>lambda</code> function notation.</li> </ul> <p>limits One of:</p> <ul style="list-style-type: none"> <li>• NULL to use the default scale values</li> <li>• A character vector that defines possible values of the scale and their order</li> <li>• A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang <code>lambda</code> function notation.</li> </ul> <p><code>na.translate</code> 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 <code>na.translate = FALSE</code>.</p> <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> uses all the levels in the factor.</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 NULL, 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>	<p>One of:</p> <ul style="list-style-type: none"> <li>• NULL 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



---

sc_dim	<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_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')
collLabels(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	<i>sc_dim_count</i>
--------------	---------------------

---

**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')
collLabels(sce) <- clusters
sce <- runUMAP(sce, assay.type = 'logcounts')
p <- sc_dim(sce, reduction = 'UMAP')
p1 <- sc_dim_count(p)
```

---

sc_dim_geom_ellipse	<i>sc_dim_geom_ellipse</i>
---------------------	----------------------------

---

**Description**

sc\_dim\_geom\_ellipse

**Usage**

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

**Arguments**

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, ...)
```

**Arguments**

geom                   geometric layer (default: geom\_text) to display the labels  
 ...                   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')
collLabels(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')
```

---

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')
collLabels(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	<i>sc_feature</i>
------------	-------------------

---

**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,  
  ...  
)
```

```
## 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,  
  ...  
)
```

```
## 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,
...
)

```

### 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.
...	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
```

---

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,
  ...
)
```

## 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,
    ...
)

## S4 method for signature 'SingleCellExperiment'
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,
  ...
)

```

### 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 rotate 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.
...	additional parameters, see also 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 mapping.</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**

ggplot object



**Examples**

```
## Not run:
library(STexampleData)
# create ExperimentHub instance
eh <- ExperimentHub()
# query STexampleData datasets
myfiles <- query(eh, "STexampleData")
spe <- myfiles[["EH7538"]]
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)

## 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,
  ...
)

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

```

)

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

```

### 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'
...	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, 7
- aes, 7
- aes (reexports), 7
- aes(), 5
- alpha, 6
  
- borders(), 5
  
- CalWkdeCpp, 3
- colour, 6
  
- DotPlot, 18
- draw\_key\_scattermore2, 4
  
- fill, 6
- fortify(), 5
  
- geom\_scattermore, 4, 10
- geom\_scattermore2, 4
- geom\_violin, 26
- ggplot(), 5
- ggplot2::discrete\_scale, 7
- ggsc (ggsc-package), 2
- ggsc-package, 2
- group, 6
- guides(), 8
  
- lambda, 8
- layer, 5
  
- rasterGrob, 5
- reexports, 7
  
- sc\_dim, 9, 15
- sc\_dim(), 11, 21
- sc\_dim, Seurat (sc\_dim), 9
- sc\_dim, Seurat-method (sc\_dim), 9
- sc\_dim, SingleCellExperiment (sc\_dim), 9
  
- sc\_dim, SingleCellExperiment-method (sc\_dim), 9
- sc\_dim\_count, 11
- sc\_dim\_geom\_ellipse, 11
- sc\_dim\_geom\_feature, 12
- sc\_dim\_geom\_label, 13
- sc\_dim\_geom\_label(), 14
- sc\_dim\_geom\_sub, 14, 15
- sc\_dim\_sub, 15
- sc\_dot, 16
- sc\_dot, Seurat (sc\_dot), 16
- sc\_dot, Seurat-method (sc\_dot), 16
- sc\_dot, SingleCellExperiment (sc\_dot), 16
- sc\_dot, SingleCellExperiment-method (sc\_dot), 16
- sc\_feature, 19
- sc\_feature(), 13, 21
- sc\_feature, Seurat (sc\_feature), 19
- sc\_feature, Seurat-method (sc\_feature), 19
- sc\_feature, SingleCellExperiment (sc\_feature), 19
- sc\_feature, SingleCellExperiment-method (sc\_feature), 19
- sc\_geom\_point, 21
- sc\_spatial, 22
- sc\_spatial, Seurat (sc\_spatial), 22
- sc\_spatial, Seurat-method (sc\_spatial), 22
- sc\_spatial, SingleCellExperiment (sc\_spatial), 22
- sc\_spatial, SingleCellExperiment-method (sc\_spatial), 22
- sc\_violin, 25
- sc\_violin, Seurat (sc\_violin), 25
- sc\_violin, Seurat-method (sc\_violin), 25
- sc\_violin, SingleCellExperiment (sc\_violin), 25
- sc\_violin, SingleCellExperiment-method (sc\_violin), 25

(sc\_violin), 25  
scale\_bg\_color\_identity  
    (scale\_bg\_color\_manual), 7  
scale\_bg\_color\_manual, 7  
scale\_bg\_colour\_discrete  
    (scale\_bg\_color\_manual), 7  
scale\_bg\_colour\_identity  
    (scale\_bg\_color\_manual), 7  
scale\_bg\_colour\_manual  
    (scale\_bg\_color\_manual), 7  
scales::pal\_hue(), 7  
shape, 6  
size, 6  
stat\_ellipse, 12  
  
theme, 7  
theme (reexports), 7  
  
x, 6  
  
y, 6