

Package ‘mascarade’

July 31, 2026

Type Package

Title Generating Cluster Masks for Single-Cell Dimensional Reduction Plots

Version 0.4.3

Description Implements a procedure to automatically generate 2D masks for clusters on dimensional reduction plots from methods like t-SNE (t-distributed stochastic neighbor embedding) or UMAP (uniform manifold approximation and projection), with a focus on single-cell RNA-sequencing data.

Imports data.table, spatstat.geom (>= 3.7-3), spatstat.explore, lifecycle, ggplot2, polyclip, polylabelr, ggforce, vctrs, rlang, cli, systemfonts (>= 1.3.0), Rcpp

LinkingTo Rcpp, BH

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Suggests testthat (>= 3.1.5), rmarkdown, knitr, patchwork, Seurat, scales, SeuratObject, svglite, colorrepel (>= 0.5.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://alserglab.github.io/mascarade/>

BugReports <https://github.com/alserglab/mascarade/issues>

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Alexey Sergushichev [aut, cre]

Maintainer Alexey Sergushichev <alsergbox@gmail.com>

Repository CRAN

Date/Publication 2026-07-30 23:30:10 UTC

Contents

exampleMascarade	2
exampleMaskTable	2
fancyMask	2
generateMask	5
generateMaskSeurat	6
geom_mark_shape	7

Index	14
--------------	-----------

exampleMascarade	<i>Example data with UMAP points from PBMC3K dataset.</i>
------------------	---

Description

The object is a list with three elements:

1. `dims` – matrix of UMAP coordinates of the cells,
2. `clusters` – vector of cell population annotations,
3. `features` – matrix with gene expression for several genes.

exampleMaskTable	<i>Example generated mask.</i>
------------------	--------------------------------

Description

Result of `generateMasks(dims=exampleMascarade$dims, clusters=exampleMascarade$clusters)`.

fancyMask	<i>Generate ggplot2 layers for a labeled cluster mask</i>
-----------	---

Description

Convenience helper that returns a list of ggplot2 components that draws polygon-like outlines and places cluster labels. The plotting limits are expanded (via `limits.expand`) to provide extra room for labels.

Usage

```
fancyMask(
  maskTable,
  ratio = NULL,
  limits.expand = ifelse(label, 0.1, 0.05),
  linewidth = 1,
  shape.expand = linewidth * unit(-1, "pt"),
  cols = "auto",
  label = TRUE,
  label.largest = TRUE,
  label.fontsize = 10,
  label.buffer = unit(2, "mm"),
  label.hardpad = unit(0, "pt"),
  label.softpad = unit(6, "pt"),
  label.fontface = "plain",
  label.margin = margin(2, 2, 2, 2, "pt"),
  label.width = NULL,
  simp_ratio = 0.001,
  con.type = "ledge"
)
```

Arguments

maskTable	A data.frame of mask coordinates. The first two columns are interpreted as x/y coordinates (in that order). Must contain at least the columns cluster (a factor) and group (grouping identifier passed to geom_mark_shape()).
ratio	Optional aspect ratio passed to ggplot2::coord_cartesian(). Use 1 for equal scaling. Default is NULL (no fixed ratio).
limits.expand	Numeric scalar giving the fraction of the x/y range to expand on both sides when setting plot limits. Default is 0.1 with labels and 0.05 with no labels.
linewidth	Line width passed to geom_mark_shape() for the outline. Default is 1.
shape.expand	Expansion or contraction applied to the marked shapes, passed to geom_mark_shape(expand = ...). Default is unit(-linewidth, "pt").
cols	Color specification for cluster outlines (and labels). One of: • "auto" (default) — inspects the plot at the time fancyMask() is added with +. If a layer maps colour to a discrete (non-numeric) variable, the mask joins that scale via aes(colour = cluster) so colours stay in sync regardless of scale_color_*() order. Otherwise (continuous colour, constant colour, or no colour aesthetic) explicit colours from scales::hue_pal() are baked in and the plot's scale system is left untouched. • "inherit" — always maps colour as an aesthetic (aes(colour = cluster)), unconditionally joining whatever colour scale is present. Useful when you want to force scale sharing; will error if the existing scale is continuous. • A palette function that accepts a single integer n and returns n colors (e.g., scales::hue_pal(), rainbow). • A single color string — applied to every cluster.

- An unnamed character vector of length equal to the number of clusters — colors are assigned to clusters in factor-level order.
- A named character vector — names must match cluster levels; order does not matter.

<code>label</code>	Boolean flag whether the labels should be displayed.
<code>label.largest</code>	Boolean flag. When TRUE (default), only the largest part of each cluster is labelled; smaller disconnected parts are drawn but not labelled. When FALSE, all parts are labelled. Ignored when <code>label = FALSE</code> .
<code>label.fontsize</code>	Label font size passed to <code>geom_mark_shape()</code> . Default is 10.
<code>label.buffer</code>	Polygon padding passed to <code>geom_mark_shape()</code> : cluster polygons are dilated by this distance and labels are kept out of the dilated zone, so each label keeps a gap from its cluster outline. Default <code>unit(2, "mm")</code> ; <code>unit(0, "mm")</code> disables.
<code>label.hardpad</code>	Hard box clearance passed to <code>geom_mark_shape()</code> : each label box is grown by this padding for all placement decisions, so labels keep at least this gap from each other. Default <code>unit(0, "pt")</code> (the label margin usually suffices; raise it mainly for <code>con.type = "box"</code>).
<code>label.softpad</code>	Extra box spacing the polish step aims for, on top of <code>label.hardpad</code> , passed to <code>geom_mark_shape()</code> . Default <code>unit(6, "pt")</code> .
<code>label.fontface</code>	Label font face passed to <code>geom_mark_shape()</code> . Default is "plain".
<code>label.margin</code>	Label margin passed to <code>geom_mark_shape()</code> . Default is <code>margin(2, 2, 2, 2, "pt")</code> .
<code>label.width</code>	Soft target width for wrapping labels, passed to <code>geom_mark_shape()</code> . A grid unit (e.g. <code>unit(30, "mm")</code>); labels are balanced across lines to keep line widths even and close to this width, without a short dangling line. NULL (default) leaves labels unwrapped. See <code>geom_mark_shape()</code> for details.
<code>simp_ratio</code>	Fraction of the polygon bounding-box area used to simplify cluster polygons before label placement: small inward (concave) vertices whose cut-off area is below <code>simp_ratio * bbox_area</code> are removed, which speeds up the box-fit and leader-routing geometry (both scale with vertex count). The simplified polygon encloses the original, so labels never overlap the real cluster. Larger values simplify more; set to 0 to disable. Default 0.001.
<code>con.type</code>	Leader / label-mark style passed to <code>geom_mark_shape()</code> : one of "ledge", "line", "box", or "none" (see the <code>geom_mark_shape()</code> Details). Default "ledge".

Details

The first two columns of `maskTable` are used as x/y coordinates. Cluster labels are taken from `maskTable$cluster`. Shapes are grouped by `maskTable$group`.

Value

A list of ggplot2 components suitable for adding to a plot with `+`, containing a `ggplot2::coord_cartesian()` specification and a `geom_mark_shape()` layer.

See Also

- `geom_mark_shape()`

Examples

```
data("exampleMascarade")
maskTable <- generateMask(dims=exampleMascarade$dims,
                          clusters=exampleMascarade$clusters)

library(ggplot2)
basePlot <- ggplot(do.call(cbind, exampleMascarade)) +
  geom_point(aes(x=UMAP_1, y=UMAP_2, color=GNLY)) +
  scale_color_gradient2(low = "#404040", high="red") +
  theme_classic()

basePlot + fancyMask(maskTable, ratio=1, cols=scales::hue_pal())
```

generateMask

Generate mask for clusters on 2D dimensional reduction plots

Description

Internally the function rasterizes and smoothes the density plots.

Usage

```
generateMask(
  dims,
  clusters,
  gridSize = 200,
  expand = 0.005,
  minDensity = lifecycle::deprecated(),
  smoothSigma = NA,
  minSize = 10,
  kernel = lifecycle::deprecated(),
  type = lifecycle::deprecated()
)
```

Arguments

<code>dims</code>	matrix of point coordinates. Rows are points, columns are dimensions. Only the first two columns are used.
<code>clusters</code>	vector of cluster annotations. Should be the same length as the number of rows in <code>dims</code> .
<code>gridSize</code>	target width and height of the raster used internally
<code>expand</code>	distance used to expand borders, represented as a fraction of $\sqrt{\text{width} \times \text{height}}$. Default: 1/200.
<code>minDensity</code>	Deprecated. Doesn't do anything.
<code>smoothSigma</code>	Deprecated. Parameter controlling smoothing and joining close cells into groups, represented as a fraction of $\sqrt{\text{width} \times \text{height}}$. Increasing this parameter can help dealing with sparse regions.

minSize	Groups of less than minSize points are ignored, unless it is the only group for a cluster
kernel	Deprecated. Doesn't do anything.
type	Deprecated. Doesn't do anything.

Value

data.table with points representing the mask borders. Each individual border line corresponds to a single level of group column. Cluster assignment is in cluster column. Within each cluster, parts are ordered by decreasing polygon area so that part index 1 is always the largest disconnected component.

Examples

```
data("exampleMascarade")
maskTable <- generateMask(dims=exampleMascarade$dims,
                          clusters=exampleMascarade$clusters)
data <- data.frame(exampleMascarade$dims,
                   cluster=exampleMascarade$clusters,
                   exampleMascarade$features)

library(ggplot2)
ggplot(data, aes(x=UMAP_1, y=UMAP_2)) +
  geom_point(aes(color=cluster)) +
  geom_path(data=maskTable, aes(group=group)) +
  coord_fixed() +
  theme_classic()
```

generateMaskSeurat	<i>Generates mask from a Seurat object. Requires SeuratObject package.</i>
--------------------	--

Description

Generates mask from a Seurat object. Requires SeuratObject package.

Usage

```
generateMaskSeurat(
  object,
  reduction = NULL,
  group.by = NULL,
  gridSize = 200,
  expand = 0.005,
  minSize = 10
)
```

Arguments

object	Seurat object
reduction	character vector specifying which reduction to use (default: DefaultDimReduc(object))
group.by	character vector specifying which field to use for clusters (default: "ident")
gridSize	target width and height of the raster used internally
expand	distance used to expand borders, represented as a fraction of sqrt(width*height). Default: 1/200.
minSize	Groups of less than minSize points are ignored, unless it is the only group for a cluster

Value

data.table with points representing the mask borders. Each individual border line corresponds to a single level of group column. Cluster assignment is in cluster column.

Examples

```
# only run if Seurat is installed
if (require("Seurat")) {
  data("pbmc_small")
  maskTable <- generateMaskSeurat(pbmc_small)

  library(ggplot2)
  # not the best plot, see vignettes for better examples
  DimPlot(pbmc_small) +
    geom_path(data=maskTable, aes(x=tSNE_1, y=tSNE_2, group=group))
}
```

geom_mark_shape

Annotate areas with polygonal shapes

Description

This geom lets you annotate sets of points via polygonal shapes. Unlike other `ggforce::geom_mark_*` functions, `geom_mark_shape` should be explicitly provided with the shape coordinates. As in `ggforce::geom_shape`, the polygon can be expanded/contracted and corners can be rounded, which is controlled by `expand` and `radius` parameters.

Usage

```
geom_mark_shape(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  expand = 0,
```

```

radius = 0,
label.margin = margin(2, 2, 2, 2, "mm"),
label.width = NULL,
label.minwidth = 0,
label.hjust = 0,
label.fontsize = 12,
label.family = "",
label.lineheight = 1,
label.fontface = c("bold", "plain"),
label.fill = "white",
label.colour = "black",
label.buffer = unit(10, "mm"),
label.hardpad = unit(0, "pt"),
label.softpad = unit(6, "pt"),
con.colour = "black",
con.size = 0.5,
con.type = "ledge",
con.linetype = 1,
con.border = "one",
con.cap = unit(3, "mm"),
con.arrow = NULL,
simp_ratio = 0.001,
...,
na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE
)

```

Arguments

- | | |
|---------|---|
| mapping | Set of aesthetic mappings created by aes() . 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 | <p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to ggplot().</p> <p>A <code>data.frame</code>, 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.</p> <p>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>).</p> |
| stat | <p>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:</p> <ul style="list-style-type: none"> • A <code>Stat</code> ggproto subclass, for example <code>StatCount</code>. |

	• 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". • 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 <code>position_jitter()</code>. 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 <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
expand	A numeric or unit vector of length one, specifying the expansion amount. Negative values will result in contraction instead. If the value is given as a numeric it will be understood as a proportion of the plot area width.
radius	As expand but specifying the corner radius.
label.margin	The margin around the annotation boxes, given by a call to <code>ggplot2::margin()</code> .
label.width	Soft target width for wrapping the label (and description). A grid unit (e.g. <code>unit(30, "mm")</code>). The text is balanced across lines so line widths are even and close to this width, avoiding a short dangling line; a line may slightly exceed it to prevent an orphan, and an over-long single word is never broken. It is a soft cap: the box shrinks to fit the wrapped text (never forced to this exact width). NULL (default) leaves the label unwrapped.
label.minwidth	The minimum width to provide for the description. If the size of the label exceeds this, the description is allowed to fill as much as the label.
label.hjust	The horizontal justification for the annotation. If it contains two elements the first will be used for the label and the second for the description.
label.fontsize	The size of the text for the annotation. If it contains two elements the first will be used for the label and the second for the description.
label.family	The font family used for the annotation. If it contains two elements the first will be used for the label and the second for the description.
label.lineheight	The height of a line as a multiplier of the fontsize. If it contains two elements the first will be used for the label and the second for the description.
label.fontface	The font face used for the annotation. If it contains two elements the first will be used for the label and the second for the description.
label.fill	The fill colour for the annotation box. Use "inherit" to use the fill from the enclosure or "inherit_col" to use the border colour of the enclosure.
label.colour	The text colour for the annotation. If it contains two elements the first will be used for the label and the second for the description. Use "inherit" to use the border colour of the enclosure or "inherit_fill" to use the fill colour from the enclosure.

label.buffer	Polygon padding: cluster polygons are dilated by this distance and labels are kept out of the dilated zone, leaving a gap between each label and its cluster outline. A grid unit; <code>unit(0, "mm")</code> disables it. Default <code>unit(10, 'mm')</code> .
label.hardpad	Hard box clearance: each label box is grown by this padding for <i>all</i> placement decisions (seed slots, label-label and label-leader conflict tests, and the polish), so labels keep at least this gap from each other. A grid unit. Defaults to <code>unit(0, 'pt')</code> – the label margin usually gives enough separation; raise it mainly for <code>con.type = 'box'</code> , where the drawn box outlines would otherwise touch.
label.softpad	Soft box spacing the polish step <i>additionally</i> aims for, on top of <code>label.hardpad</code> (it does not tighten the hard conflict tests). A grid unit. Default <code>unit(6, 'pt')</code> .
con.colour	The colour for the line connecting the annotation to the mark. Use <code>"inherit"</code> to use the border colour of the enclosure or <code>"inherit_fill"</code> to use the fill colour from the enclosure.
con.size	The width of the connector. Use <code>"inherit"</code> to use the border width of the enclosure.
con.type	Leader / label-mark style: one of <code>"ledge"</code> , <code>"line"</code> , <code>"box"</code> , or <code>"none"</code> (see Details). Default <code>"ledge"</code> .
con.linetype	The linetype of the connector. Use <code>"inherit"</code> to use the border linetype of the enclosure.
con.border	The bordertype of the connector. Either <code>"one"</code> (to draw a line on the horizontal side closest to the mark), <code>"all"</code> (to draw a border on all sides), or <code>"none"</code> (not going to explain that one).
con.cap	The distance before the mark that the line should stop at.
con.arrow	An arrow specification for the connection using <code>grid::arrow()</code> for the end pointing towards the mark.
simp_ratio	Fraction of the polygon bounding-box area used to simplify cluster polygons before label placement (removes small inward vertices; the simplified polygon encloses the original, so labels never overlap the real shape). Speeds up placement geometry. Larger values simplify more; 0 disables. Default 0.001.
...	Other arguments passed on to <code>layer()</code> 's <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored. • Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data. • When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept.

	• Inversely, when constructing a layer using a <code>geom_*()</code> function, the <code>...</code> argument can be used to pass on parameters to the <code>stat</code> part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The <code>stat</code>'s documentation lists which parameters it can accept. • The <code>key_glyph</code> argument of <code>layer()</code> may also be passed on through <code>...</code>. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<code>show.legend</code>	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.
<code>inherit.aes</code>	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. <code>borders()</code> .

Details

`con.type` selects how each label is connected to its cluster:

- "ledge" — leader from the box corner facing the cluster, plus a short horizontal ledge along the box edge at the leader's start (the default).
- "line" — leader from the box corner or edge-midpoint facing the cluster, with no ledge.
- "box" — placed as for "line", and additionally outlines the label's bounding box.
- "none" — no connector is drawn; the label is still placed as for "line".

Value

A `ggplot2` layer (`ggplot2::layer`) that adds polygonal shape annotations to a plot.

Aesthetics

`geom_mark_shape` understand the following aesthetics (required aesthetics are in bold):

- **x**
- **y**
- filter
- label
- description
- color
- fill
- group
- size
- linetype
- alpha

Annotation

All `geom_mark_*` allow you to put descriptive textboxes connected to the mark on the plot, using the `label` and `description` aesthetics. The textboxes are automatically placed close to the mark, but without obscuring any of the datapoints in the layer. The placement is dynamic so if you resize the plot you'll see that the annotation might move around as areas become big enough or too small to fit the annotation. If there's not enough space for the annotation without overlapping data it will not get drawn. In these cases try resizing the plot, change the size of the annotation, or decrease the buffer region around the marks.

Filtering

Often marks are used to draw attention to, or annotate specific features of the plot and it is thus not desirable to have marks around everything. While it is possible to simply pre-filter the data used for the mark layer, the `geom_mark_*` geoms also comes with a dedicated `filter` aesthetic that, if set, will remove all rows where it evaluates to `FALSE`. There are multiple benefits of using this instead of prefiltering. First, you don't have to change your data source, making your code more adaptable for exploration. Second, the data removed by the `filter` aesthetic is remembered by the geom, and any annotation will take care not to overlap with the removed data.

Examples

```
library(ggplot2)
shape1 <- data.frame(
  x = c(0, 3, 3, 2, 2, 1, 1, 0),
  y = c(0, 0, 3, 3, 1, 1, 3, 3),
  label = "U-shape",
  description = "two prongs on a base"
)
shape2 <- data.frame(
  x = c(0, 3, 3, 0)+4,
  y = c(0, 0, 3, 3),
  label = "square",
  description = "four equal sides"
)
shape3 <- data.frame(
  x = c(0, 1.5, 3, 1.5)+8,
  y = c(1.5, 0, 1.5, 3),
  label = "diamond",
  description = "a square on its corner"
)
shapes <- rbind(shape1, shape2, shape3)

# Label only
ggplot(shapes, aes(x=x, y=y, label=label, color=label, fill=label)) +
  geom_mark_shape() +
  ylim(0, 5)

# Label with a secondary description line
ggplot(shapes, aes(x=x, y=y, label=label, description=description,
  color=label, fill=label)) +
  geom_mark_shape() +
```

geom_mark_shape

13

`ylim(0, 5)`

Index

* mark geoms

geom_mark_shape, [7](#)

aes(), [8](#)

borders(), [11](#)

exampleMascarade, [2](#)

exampleMaskTable, [2](#)

fancyMask, [2](#)

fortify(), [8](#)

generateMask, [5](#)

generateMaskSeurat, [6](#)

geom_mark_shape, [7](#)

ggplot(), [8](#)

ggplot2::margin(), [9](#)

grid::arrow(), [10](#)

key glyphs, [11](#)

layer position, [9](#)

layer stat, [9](#)

layer(), [10](#), [11](#)