Package: mascarade (via r-universe)

May 23, 2026

Type Package

Title Generating Cluster Masks for Single-Cell Dimensional Reduction Plots

Version 0.3.4.9000

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, ggforce, vctrs, rlang, cli, systemfonts

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 3.5)

Suggests testthat (>= 3.0.0), rmarkdown, knitr, patchwork, ggsci, Seurat, scales, SeuratObject, svglite

Config/testthat/edition 3

VignetteBuilder knitr

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

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

Roxygen list(markdown=TRUE)

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev

Repository https://alserglab.r-universe.dev

Date/Publication 2026-04-23 15:27:55 UTC

RemoteUrl https://github.com/alserglab/mascarade

RemoteRef HEAD

RemoteSha 2b98742812121513f4dee056afd4b4f1416d6acf

Contents

`exampleMascarade` *Example data with UMAP points from PBMC3K dataset.*

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 withgene expression for several genes.

`fancyMask` *Generate ggplot2 layers for a labeled cluster mask*

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(0, "cm"),
  label.fontface = "plain",
  label.margin = margin(2, 2, 2, 2, "pt"),
  simp_ratio = 0.001
)
```

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.
label	Boolean flag whether the labels should be displayed.
label.largest	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 `label = FALSE`.
label.fontsize	Label font size passed to `geom_mark_shape()`. Default is `10`.
label.buffer	Label buffer distance passed to `geom_mark_shape()`. Default is `unit(0, "cm")`.
label.fontface	Label font face passed to `geom_mark_shape()`. Default is `"plain"`.
label.margin	Label margin passed to `geom_mark_shape()`. Default is `margin(2, 2, 2, 2, "pt")`.
simp_ratio	Fraction of the polygon bounding box area used as the label-placement simplification threshold. Cluster polygons are simplified before the label placement search by removing small concave vertices, which reduces computation while guaranteeing the simplified polygon encloses the original. Larger values simplify more aggressively; set to `0` to disable. Default is `0.001`.

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

dims	matrix of point coordinates. Rows are points, columns are dimensions. Only the first two columns are used.
clusters	vector of cluster annotations. Should be the same length as the number of rows in dims.
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.
minDensity	Deprecated. Doesn't do anything.
smoothSigma	Deprecated. Parameter controlling smoothing and joining close cells into groups, represented as a fraction of sqrt(width*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 *Generates mask from a Seurat object. Requires* SeuratObject *package.*

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 = unit(50, "mm"),
  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"),
  con.colour = "black",
  con.size = 0.5,
  con.type = "elbow",
  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 inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

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

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

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

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

`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 `ggplot2::margin()`.

`label.width` A fixed width for the label. Set to `NULL` to let the text or `label.minwidth` decide.

`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 multipler 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` The size of the region around the mark where labels cannot be placed.

`con.colour` The colour for the line connecting the annotation to the mark. Use `"inherit"` to use the border colour of the enclosure or `"inherit_fill"` to use the fill colour from the enclosure.

`con.size` The width of the connector. Use `"inherit"` to use the border width of the enclosure.

`con.type` The type of the connector. Either `"elbow"`, `"straight"`, or `"none"`.

`con.linetype` The linetype of the connector. Use `"inherit"` to use the border linetype of the enclosure.

`con.border` The bordertype of the connector. Either `"one"` (to draw a line on the horizontal side closest to the mark), `"all"` (to draw a border on all sides), or `"none"` (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 `grid::arrow()` for the end pointing towards the mark.

`simp_ratio` Fraction of the polygon bounding box area used as the label-placement simplification threshold. Cluster polygons are simplified before the label placement search by removing small concave vertices, which reduces computation while guaranteeing the simplified polygon encloses the original. Larger values simplify more aggressively; set to `0` to disable. Default is `0.001`.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* 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, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. 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 `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`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. borders().

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**
- x0 *(used to anchor the label)*
- y0 *(used to anchor the label)*
- 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 evalutates 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="bracket"
)
shape2 <- data.frame(
    x = c(0, 3, 3, 0)+4,
    y = c(0, 0, 3, 3),
    label="square"
)
shape3 <- data.frame(
    x = c(0, 1.5, 3, 1.5)+8,
    y = c(1.5, 0, 1.5, 3),
    label="diamond"
)

ggplot(rbind(shape1, shape2, shape3), aes(x=x, y=y, label=label, color=label, fill=label)) +
    geom_mark_shape() +
    ylim(0, 5)
```

Index