

# Package ‘GOfan’

May 4, 2026

**Type** Package

**Title** Sunburst Plot for Enriched Gene Ontology Terms

**Version** 1.1.0

**Description** GOfan provides an intuitive and compact visualization of Gene Ontology (GO) enrichment results using a sunburst layout inspired by SynGO, preserving hierarchical relationships among GO terms and allowing color-based encoding of information such as p-values or gene counts. By converting complex GO DAGs into clean, circular representations, it allows researchers to quickly grasp the hierarchical structure and biological significance of enriched terms. The interactive and customizable visualizations facilitate exploration of key GO categories, enhancing interpretation and presentation of enrichment analyses.

**LazyData** false

**License** GPL-3

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**VignetteBuilder** knitr

**biocViews** Visualization, GO, Annotation

**Depends** R (>= 4.5.0), ggplot2

**Imports** AnnotationDbi, grid, grDevices, GO.db, igraph, methods, plotly, rlang, stats, scales, vctrs

**Suggests** BiocStyle, knitr, rmarkdown, testthat, org.Dr.eb.db

**URL** <https://github.com/jianhong/GOfan>

**BugReports** <https://github.com/jianhong/GOfan/issues>

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

**git\_branch** devel

**git\_last\_commit** 544d98d

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-03

**Author** Jianhong Ou [aut, cre] (ORCID: <<https://orcid.org/0000-0002-8652-2488>>),  
Kenneth Poss [aut, fnd]

**Maintainer** Jianhong Ou <jou@morgridge.org>

## Contents

GOfan-package . . . . .	2
filterGraph . . . . .	2
geom_sunburst . . . . .	3
getAncestors . . . . .	7
getGOalias . . . . .	7
getGraph . . . . .	8
ggSunburst . . . . .	8
goGraph . . . . .	10
simplifyDAG . . . . .	10
sunburstGO . . . . .	11
<b>Index</b>	<b>13</b>

---

GOfan-package	<i>Sunburst Plot for Enriched Gene Ontology Terms</i>
---------------	---

---

### Description

GOfan provides a simple and intuitive way to visualize Gene Ontology (GO) enrichment results using a sunburst layout inspired by SynGO. Unlike graph-based or dot plot methods, it preserves hierarchical relationships among GO terms while maintaining a clean, interpretable view. GOfan accepts any enrichment results containing GO identifiers and uses color to represent additional information such as p-values or gene counts, making GO analysis both informative and visually engaging.

### Author(s)

Jianhong Ou

Maintainer: Jianhong Ou [jianhong.ou@morgridge.org](mailto:jianhong.ou@morgridge.org)

### See Also

Useful links:

- <https://github.com/jianhong/GOfan>
- Report bugs at <https://github.com/jianhong/GOfan/issues>

---

filterGraph	<i>Cut the enriched GO terms by the distances from the root after simplify</i>
-------------	--

---

### Description

Cut the input igraph object by the distances from the root.

**Usage**

```
filterGraph(
  g,
  leaveTerms,
  cutoff = 4,
  filterNodesByEdgeNumber = 0,
  mustkeep = c(),
  onlyKeep = c()
)
```

**Arguments**

g	A igraph object
leaveTerms	Leaves must contained GO terms.
cutoff	The cutoff distance from the root
filterNodesByEdgeNumber	Filter the graphs by the edge number.
mustkeep	The GO terms must be kept.
onlyKeep	Only keep branches with give GO terms.

**Value**

A igraph object after filtering

**Examples**

```
library(igraph)
g_gnp <- sample_gnp(n = 25, p = 0.05)
filterGraph(g_gnp, leaveTerms = V(g_gnp), cutoff = 2)
```

---

geom_sunburst	<i>Sunburst plot</i>
---------------	----------------------

---

**Description**

create a sunburst geom.

**Usage**

```
geom_sunburst(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  lineend = "butt",
  linejoin = "mitre",
  parse = FALSE,
  size.unit = "mm",
```

```

  check_overlap = FALSE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

```

## Arguments

mapping	Set of aesthetic mappings created by <code>ggplot2::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 <code>mapping</code> 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 <code>ggplot2::ggplot()</code>.</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 <code>ggplot2::fortify()</code> 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"> <li>• A Stat 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	<p>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:</p> <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>
...	<p>Other arguments passed on to <code>ggplot2::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.</p> <ul style="list-style-type: none"> <li>• 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 <b>Aesthetics</b> 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</li> </ul>

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 `ggplot2::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.

<code>lineend</code>	Line end style (round, butt, square).
<code>linejoin</code>	Line join style (round, mitre, bevel).
<code>parse</code>	If TRUE, the labels will be parsed into expressions and displayed as described in <code>?plotmath</code> .
<code>size.unit</code>	How the size aesthetic is interpreted: as millimetres ("mm", default), points ("pt"), centimetres ("cm"), inches ("in"), or picas ("pc").
<code>check_overlap</code>	If TRUE, text that overlaps previous text in the same layer will not be plotted. <code>check_overlap</code> happens at draw time and in the order of the data. Therefore data should be arranged by the label column before calling <code>geom_text()</code> . Note that this argument is not supported by <code>geom_label()</code> .
<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. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.
<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>ggplot2::annotation_borders</code>

## Details

Please note that the width and height aesthetics are not true position aesthetics and therefore are not subject to scale transformation. It is only after transformation that these aesthetics are applied.

## Value

A `ggplot` object

## Aesthetics

`geom_sunburst` understands the following aesthetics:

- `x` define the locations of `x`.
- `y` define the locations of `y`.
- `xmin` define the bottom of rectangle

- ymin define the left of rectangle
- xmax define the top of rectangle
- ymax define the right of rectangle
- width define the width of rectangle
- height define the height of rectangle
- colour rectangle border color
- fill rectangle fill color
- alpha rectangle fill alpha and font alpha
- linewidth line width for rectangle
- linetype line type for rectangle
- label label text
- angle label angle
- family label family
- size label default size
- fontcolour font color
- fontface font face
- lineheight font line height
- hjust horizontal just for label
- vjust vertical just for label
- rotate90 Rotate the labels 90 degree or not. Default NULL will try to auto rotate the labels according the space.
- sub\_rect A proportional sub-rectangle representing a share of the whole.

### Examples

```
plotdata <- data.frame(
  id = c("GO:0023052", "GO:0007267", "GO:0099536", "GO:0099537", "GO:0098916"),
  x = 0.5,
  y = seq.int(5),
  xmin = 0,
  ymin = c(0.5, 1.5, 2.5, 3.5, 4.5),
  xmax = 1,
  ymax = c(1.5, 2.5, 3.5, 4.5, 5.5),
  fill = seq(1, 5),
  label = c(
    "signaling", "cell-cell signaling", "synaptic signaling",
    "trans-synaptic signaling", "anterograde trans-synaptic signaling"
  )
)
library(ggplot2)
ggplot(plotdata, aes(
  x = x, y = y, xmin = xmin, ymin = ymin, xmax = xmax, ymax = ymax,
  fill = fill, label = label
)) +
  geom_sunburst(size = 0.5, angle = -90) +
  coord_polar()
```

---

getAncestors	<i>Recursive function to extract (ancestor, offspring) pairs</i>
--------------	--

---

**Description**

Recursive function to extract (ancestor, offspring) pairs

**Usage**

```
getAncestors(GO_IDs, onto = c("BP", "CC", "MF"))
```

**Arguments**

GO_IDs	The Gene ontology term ids
onto	The category of the GO ids. It should be one of "BP", "CC", or "MF".

**Value**

A data frame with columns ancestor and offspring

**Examples**

```
ids <- c(
  "GO:0099536", "GO:0099537", "GO:0007268",
  "GO:0098916", "GO:0050804", "GO:0099177"
)
df <- getAncestors(ids, onto = "BP")
head(df)
```

---

getGOalias	<i>Extract all the entrez IDs from a given GO IDs</i>
------------	---

---

**Description**

Extract all the entrez IDs from a given GO IDs

**Usage**

```
getGOalias(GO_IDs, org)
```

**Arguments**

GO_IDs	The Gene ontology term id
org	The OrgDb

**Value**

A list of entrez IDs for the given GO IDs

**Examples**

```
library(org.Dr.eg.db)
ids <- c(
  "GO:0099536", "GO:0099537", "GO:0007268",
  "GO:0098916", "GO:0050804", "GO:0099177"
)
eids <- getGOalias(ids, org.Dr.eg.db)
```

---

getGraph

*Prepare the graph for Sunburst plot*


---

**Description**

By a given GO enrichment results, extract GO ancestor and offspring info from GO.db and then generate a simplified tree like graph.

**Usage**

```
getGraph(df, org, termID = "ID", onto = c("BP", "CC", "MF"))
```

**Arguments**

df	A data frame with enriched GO terms
org	An Go3AnnDbBimap object eg org.Dr.egGO2ALLEGs
termID	Column name in df which store the GO IDs
onto	The ontology category of the GO IDs

**Value**

A igraph graph.

**Examples**

```
library(org.Dr.eg.db)
goids <- c("GO:0099536", "GO:0099537", "GO:0007268", "GO:0098916", "GO:0050804")
g <- getGraph(data.frame(ID = goids), org = org.Dr.eg.db, onto = "BP")
```

---

ggSunburst

*Creates sunburst diagram using ggplot2.*


---

**Description**

Creates sunburst diagram using ggplot2.

**Usage**

```
ggSunburst(
  plotdata,
  fontsize = 1,
  rotate90 = NULL,
  maxCharacters = 30,
  legendTitle = "color",
  start = 0,
  end = NULL,
  clip = "off",
  expand = FALSE,
  ...
)
```

**Arguments**

plotdata	A data.frame.
fontsize	Default fontsize.
rotate90	Rotate the labels 90 degree or not. Default NULL will try to auto rotate the labels according the space.
maxCharacters	Maximal number of characters for labels
legendTitle	The title of the legend.
start, end	Offset of starting or ending point from 12 o'clock in radians. see <a href="#">coord_radial</a> .
clip	Should drawing be clipped to the extent of the plot panel? Default "on" means yes.
expand	If TRUE, adds a small expansion factor the the limits to prevent overlap between data and axes. If FALSE, the default, limits are taken directly from the scale.
...	Other parameters (except theta) passed to <a href="#">coord_radial</a> .

**Value**

A [ggplot](#) object

**Examples**

```
plotdata <- data.frame(
  id = c("GO:0023052", "GO:0007267", "GO:0099536", "GO:0099537", "GO:0098916"),
  x = 0.5,
  y = seq.int(5),
  xmin = 0,
  ymin = c(0.5, 1.5, 2.5, 3.5, 4.5),
  xmax = 1,
  ymax = c(1.5, 2.5, 3.5, 4.5, 5.5),
  fill = seq(1, 5),
  label = c(
    "signaling", "cell-cell signaling", "synaptic signaling",
    "trans-synaptic signaling", "anterograde trans-synaptic signaling"
  )
)
ggSunburst(plotdata, end = pi / 2)
```

---

`goGraph`*Creating igraph graphs from ancestor\_offspring data frame*

---

**Description**

This function creates an igraph graph from one data frames containing the ancestor and offspring information.

**Usage**

```
goGraph(df)
```

**Arguments**

`df` A data frame, output of `getAncestors`

**Value**

A igraph graph.

**Examples**

```
goids <- c("GO:0099536", "GO:0099537", "GO:0007268", "GO:0098916", "GO:0050804")
anc <- getAncestors(goids, onto = "BP")
goGraph(anc)
```

---

`simplifyDAG`*Simplify DAG by keeping strongest parent in sub-graphs*

---

**Description**

simplify graph by keeping only strongest parent, to make the DAG to a tree like graph

**Usage**

```
simplifyDAG(g, org)
```

**Arguments**

`g` A igraph object  
`org` A Go3AnnDbBimap object

**Value**

Simplified graph

**Examples**

```

library(igraph)
library(org.Dr.eg.db)
edges <- data.frame(
  ancestor = c("GO:0007154", "GO:0007267", "GO:0099536", "GO:0099537"),
  offspring = c("GO:0099536", "GO:0099536", "GO:0099537", "GO:0098916")
)
g <- graph_from_data_frame(edges)
g1 <- simplifyDAG(g, org.Dr.eg.db)

```

sunburstGO

*Sunburst plot for enriched GO term***Description**

Sunburst plot for enriched GO term

**Usage**

```

sunburstGO(
  df,
  org,
  g,
  termID = "ID",
  fill = "qvalue",
  sub_rect = NULL,
  GO_annotation_level_cutoff = 4,
  filterNodesByEdgeNumber = 2,
  mustkeep = c(),
  onlyKeep = c(),
  fillNAbby0 = TRUE,
  onto = c("BP", "CC", "MF"),
  plotBy = c("plotly", "ggplot2"),
  ...
)

```

**Arguments**

df	A data frame with enriched GO terms
org	An OrgDb object
g	An igraph graph. Output of <a href="#">getGraph</a> .
termID	Column name in df which store the GO IDs
fill	Column name in df used to set the fill colors
sub_rect	Column name in df used to set the area of a proportional sub-rectangle, which represent a share of the whole. The values should be a number in the range from 0 to 1. If it is a count number, it will be convert to a proportion by divided the total number of features in the term. Otherwise, will simply re-scale to .
GO_annotation_level_cutoff	The cutoff of the GO annotation levels

<code>filterNodesByEdgeNumber</code>	Filter the sub graphs by the edge numbers.
<code>mustkeep</code>	The GO terms must be kept.
<code>onlyKeep</code>	Only keep branches with give GO terms.
<code>fillNAby0</code>	Fill the NA values by 0 or not for the color column.
<code>onto</code>	The ontology category of the GO IDs
<code>plotBy</code>	plot tools, plotly or ggplot2.
<code>...</code>	parameter passed to <a href="#">ggSunburst</a> .

**Value**

A plot handle

**Examples**

```
library(org.Dr.eg.db)
df <- data.frame(
  ID = c("GO:0007267", "GO:0099536", "GO:0099537", "GO:0098916"),
  qvalue = -10 * log10(runif(4, max = 0.05))
)
sunburstGO(df, org.Dr.eg.db)
```

# Index

- \* **package**
  - GOfan-package, 2
- coord\_radial, 9
- filterGraph, 2
- geom\_sunburst, 3
- getAncestors, 7, 10
- getGOalias, 7
- getGraph, 8, 11
- ggplot, 5, 9
- ggplot2::aes(), 4
- ggplot2::annotation\_borders(), 5
- ggplot2::fortify(), 4
- ggplot2::ggplot(), 4
- ggplot2::layer(), 4, 5
- ggSunburst, 8, 12
- GOfan (GOfan-package), 2
- GOfan-package, 2
- goGraph, 10
- key glyphs, 5
- layer position, 4
- layer stat, 4
- simplifyDAG, 10
- sunburstGO, 11