

# Package ‘EnhancedVolcano’

March 10, 2025

**Type** Package

**Title** Publication-ready volcano plots with enhanced colouring and labeling

**Version** 1.25.0

**Maintainer** Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

**Description** Volcano plots represent a useful way to visualise the results of differential expression analyses. Here, we present a highly-configurable function that produces publication-ready volcano plots. EnhancedVolcano will attempt to fit as many point labels in the plot window as possible, thus avoiding 'clogging' up the plot with labels that could not otherwise have been read. Other functionality allows the user to identify up to 4 different types of attributes in the same plot space via colour, shape, size, and shade parameter configurations.

**License** GPL-3

**Depends** ggplot2, ggrepel

**Imports** methods

**Suggests** ggalt, ggrastr, RUnit, BiocGenerics, knitr, DESeq2, pasilla, airway, org.Hs.eg.db, gridExtra, magrittr, rmarkdown

**URL** <https://github.com/kevinblighe/EnhancedVolcano>

**biocViews** RNASeq, GeneExpression, Transcription, DifferentialExpression, ImmunoOncology

**VignetteBuilder** knitr

**RoxygenNote** 7.1.2

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

**git\_branch** devel

**git\_last\_commit** 88a8e6a

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

**Repository** Bioconductor 3.21

**Date/Publication** 2025-03-10

**Author** Kevin Blighe [aut, cre],  
Sharmila Rana [aut],  
Emir Turkes [ctb],

Benjamin Ostendorf [ctb],  
 Andrea Grioni [ctb],  
 Myles Lewis [aut]

## Contents

EnhancedVolcano . . . . .	2
<b>Index</b>	<b>9</b>

---

EnhancedVolcano	<i>Publication-ready volcano plots with enhanced colouring and labeling.</i>
-----------------	------------------------------------------------------------------------------

---

## Description

Volcano plots represent a useful way to visualise the results of differential expression analyses. Here, we present a highly-configurable function that produces publication-ready volcano plots [`@EnhancedVolcano`]. `EnhancedVolcano` will attempt to fit as many variable names in the plot window as possible, thus avoiding 'clogging' up the plot with labels that could not otherwise have been read.

## Usage

```
EnhancedVolcano(
  toptable,
  lab,
  x,
  y,
  selectLab = NULL,
  xlim = c(min(toptable[[x]], na.rm = TRUE) - 1.5, max(toptable[[x]], na.rm = TRUE) +
    1.5),
  ylim = c(0, max(-log10(toptable[[y]]), na.rm = TRUE) + 5),
  xlab = bquote(~Log[2] ~ "fold change"),
  ylab = bquote(~-Log[10] ~ italic(P)),
  axisLabSize = 18,
  title = "Volcano plot",
  subtitle = bquote(italic(EnhancedVolcano)),
  caption = paste0("total = ", nrow(toptable), " variables"),
  titleLabSize = 18,
  subtitleLabSize = 14,
  captionLabSize = 14,
  pCutoff = 1e-05,
  pCutoffCol = y,
  FCcutoff = 1,
  cutoffLineType = "longdash",
  cutoffLineCol = "black",
```

```
cutoffLineWidth = 0.4,
pointSize = 2,
labSize = 5,
labCol = "black",
labFace = "plain",
boxedLabels = FALSE,
parseLabels = FALSE,
shape = 19,
shapeCustom = NULL,
col = c("grey30", "forestgreen", "royalblue", "red2"),
colCustom = NULL,
colAlpha = 1/2,
colGradient = NULL,
colGradientBreaks = c(pCutoff, 1),
colGradientLabels = c("0", "1.0"),
colGradientLimits = c(0, 1),
legendLabels = c("NS", expression(Log[2] ~ FC), "p-value", expression(p - value ~ and
  ~ log[2] ~ FC)),
legendPosition = "top",
legendLabSize = 14,
legendIconSize = 5,
legendDropLevels = TRUE,
encircle = NULL,
encircleCol = "black",
encircleFill = "pink",
encircleAlpha = 3/4,
encircleSize = 2.5,
shade = NULL,
shadeFill = "grey",
shadeAlpha = 1/2,
shadeSize = 0.01,
shadeBins = 2,
drawConnectors = FALSE,
widthConnectors = 0.5,
typeConnectors = "closed",
endsConnectors = "first",
lengthConnectors = unit(0.01, "npc"),
colConnectors = "grey10",
max.overlaps = 15,
maxoverlapsConnectors = NULL,
min.segment.length = 0,
directionConnectors = "both",
arrowheads = TRUE,
hline = NULL,
hlineType = "longdash",
hlineCol = "black",
hlineWidth = 0.4,
vline = NULL,
```

```

vlineType = "longdash",
vlineCol = "black",
vlineWidth = 0.4,
gridlines.major = TRUE,
gridlines.minor = TRUE,
border = "partial",
borderWidth = 0.8,
borderColour = "black",
raster = FALSE
)

```

### Arguments

toptable	A data-frame of test statistics (if not, a data frame, an attempt will be made to convert it to one). Requires at least the following: column for variable names (can be rownames); a column for log2 fold changes; a column for nominal or adjusted p-value.
lab	A column name in toptable containing variable names. Can be rownames(toptable).
x	A column name in toptable containing log2 fold changes.
y	A column name in toptable containing nominal or adjusted p-values.
selectLab	A vector containing a subset of lab.
xlim	Limits of the x-axis.
ylim	Limits of the y-axis.
xlab	Label for x-axis.
ylab	Label for y-axis.
axisLabSize	Size of x- and y-axis labels.
title	Plot title.
subtitle	Plot subtitle.
caption	Plot caption.
titleLabSize	Size of plot title.
subtitleLabSize	Size of plot subtitle.
captionLabSize	Size of plot caption.
pCutoff	Cut-off for statistical significance. A horizontal line will be drawn at $-\log_{10}(\text{pCutoff})$ .
pCutoffCol	Column name of statistical significance values to be used as the cut-off. A typical usage situation would be to pass nominal [un-adjusted] p-values as 'y', but adjusted p-values as pCutoffCol. In this way, a plot is generated via $-\log_{10}(\text{unadjusted p-value})$ , but cut-offs based on adjusted p-values.
FCcutoff	Cut-off for absolute log2 fold-change. Vertical lines will be drawn at the negative and positive values of $\log_2 \text{FCcutoff}$ .
cutoffLineType	Line type for FCcutoff and pCutoff ('blank', 'solid', 'dashed', 'dotted', 'dot-dash', 'longdash', 'twodash').
cutoffLineCol	Line colour for FCcutoff and pCutoff.

<code>cutoffLineWidth</code>	Line width for <code>FCcutoff</code> and <code>pCutoff</code> .
<code>pointSize</code>	Size of plotted points for each variable. Can be a single value or a vector of sizes.
<code>labSize</code>	Size of labels for each variable.
<code>labCol</code>	Colour of labels for each variable.
<code>labFace</code>	Font face of labels for each variable.
<code>boxedLabels</code>	Logical, indicating whether or not to draw labels in boxes.
<code>parseLabels</code>	Logical, indicating whether or not to parse expressions in labels
<code>shape</code>	Shape of the plotted points. Either a single value for all points, or 4 values corresponding to the default 4 legend labels specified by <code>legendLabels</code> .
<code>shapeCustom</code>	Named vector / key-value pairs that will over-ride the default shape scheme. The order must match that of <code>toptable</code> . Names / keys relate to groups / categories; values relate to shape encodings.
<code>col</code>	Colour shading for plotted points, corresponding to the default 4 legend labels specified by <code>legendLabels</code> .
<code>colCustom</code>	Named vector / key-value pairs that will over-ride the default colour scheme. The order must match that of <code>toptable</code> . Names / keys relate to groups / categories; values relate to colour.
<code>colAlpha</code>	Alpha for purposes of controlling colour transparency of variable points.
<code>colGradient</code>	If activated, over-rides the default discrete colour scheme and replaces it with a continous scheme that shades based on nominal or adjusted p-value specified by <code>y</code> . For example, <code>c('red2', 'blue2')</code> .
<code>colGradientBreaks</code>	Break-points for the two colours specified by <code>colGradient</code> .
<code>colGradientLabels</code>	Labels for the break-points specified by <code>colGradientBreaks</code> .
<code>colGradientLimits</code>	Limits of the colour scheme specified by <code>colGradient</code> , i.e., max and min possible p-values.
<code>legendLabels</code>	Plot legend text labels.
<code>legendPosition</code>	Position of legend ('top', 'bottom', 'left', 'right').
<code>legendLabSize</code>	Size of plot legend text.
<code>legendIconSize</code>	Size of plot legend icons / symbols.
<code>legendDropLevels</code>	Logical, drop unused factor levels from legend.
<code>encircle</code>	A vector of variable names to encircle. Requires installation of package <a href="#">ggalt</a> .
<code>encircleCol</code>	Colour of the encircled line.
<code>encircleFill</code>	Colour fill of the encircled region.
<code>encircleAlpha</code>	Alpha for purposes of controlling colour transparency of encircled region.
<code>encircleSize</code>	Line width of the encircled line.

<code>shade</code>	A vector of variable names to shade.
<code>shadeFill</code>	Colour of shaded regions.
<code>shadeAlpha</code>	Alpha for purposes of controlling colour transparency of shaded region.
<code>shadeSize</code>	Size of the shade contour lines.
<code>shadeBins</code>	Number of bins for the density of the shade.
<code>drawConnectors</code>	Logical, indicating whether or not to connect plot labels to their corresponding points by line connectors.
<code>widthConnectors</code>	Line width of connectors.
<code>typeConnectors</code>	Have the arrow head open ('open') or filled ('closed')?
<code>endsConnectors</code>	Which end of connectors to draw arrow head? ('last', 'first', 'both').
<code>lengthConnectors</code>	Length (size) of the connector arrowheads.
<code>colConnectors</code>	Line colour of connectors and line segments.
<code>max.overlaps</code>	Equivalent of <code>max.overlaps</code> in <code>ggrepel</code> . Set to 'Inf' to always display all labels when <code>drawConnectors = TRUE</code> .
<code>maxoverlapsConnectors</code>	See <code>max.overlaps</code> .
<code>min.segment.length</code>	When <code>drawConnectors = TRUE</code> , specifies the minimum length of the connector line segments.
<code>directionConnectors</code>	direction in which to draw connectors. 'both', 'x', or 'y'.
<code>arrowheads</code>	Logical, indicating whether or not to draw arrow heads or or just have straight lines.
<code>hline</code>	Draw one or more horizontal lines passing through this/these values on y-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., <code>c(60,90)</code> .
<code>hlineType</code>	Line type for <code>hline</code> ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').
<code>hlineCol</code>	Colour of <code>hline</code> .
<code>hlineWidth</code>	Width of <code>hline</code> .
<code>vline</code>	Draw one or more vertical lines passing through this/these values on x-axis. For single values, only a single numerical value is necessary. For multiple lines, pass these as a vector, e.g., <code>c(60,90)</code> .
<code>vlineType</code>	Line type for <code>vline</code> ('blank', 'solid', 'dashed', 'dotted', 'dotdash', 'longdash', 'twodash').
<code>vlineCol</code>	Colour of <code>vline</code> .
<code>vlineWidth</code>	Width of <code>vline</code> .
<code>gridlines.major</code>	Logical, indicating whether or not to draw major gridlines.

<code>gridlines.minor</code>	Logical, indicating whether or not to draw minor gridlines.
<code>border</code>	Add a border for just the x and y axes ('partial') or the entire plot grid ('full')?
<code>borderWidth</code>	Width of the border on the x and y axes.
<code>borderColour</code>	Colour of the border on the x and y axes.
<code>raster</code>	Logical, indicating whether to rasterize the <code>geom_point</code> layer. Requires installation of <a href="#">ggrastr</a> .

## Details

Volcano plots represent a useful way to visualise the results of differential expression analyses. Here, we present a highly-configurable function that produces publication-ready volcano plots [[@EnhancedVolcano](#)]. `EnhancedVolcano` will attempt to fit as many variable names in the plot window as possible, thus avoiding 'clogging' up the plot with labels that could not otherwise have been read.

## Value

A `ggplot2` object.

## Author(s)

Kevin Blighe <[kevin@clinicalbioinformatics.co.uk](mailto:kevin@clinicalbioinformatics.co.uk)>

## Examples

```
library('pasilla')
pasCts <- system.file('extdata', 'pasilla_gene_counts.tsv',
  package='pasilla', mustWork=TRUE)
pasAnno <- system.file('extdata', 'pasilla_sample_annotation.csv',
  package='pasilla', mustWork=TRUE)
cts <- as.matrix(read.csv(pasCts, sep='\t', row.names='gene_id'))
coldata <- read.csv(pasAnno, row.names=1)
coldata <- coldata[,c('condition', 'type')]
rownames(coldata) <- sub('fb', '', rownames(coldata))
cts <- cts[, rownames(coldata)]
library('DESeq2')
dds <- DESeqDataSetFromMatrix(countData = cts,
  colData = coldata,
  design = ~ condition)

featureData <- data.frame(gene=rownames(cts))
mcols(dds) <- DataFrame(mcols(dds), featureData)
dds <- DESeq(dds)
res <- results(dds)

EnhancedVolcano(res,
  lab = rownames(res),
  x = 'log2FoldChange',
  y = 'pvalue',
```

```
pCutoff = 10e-4,  
FCcutoff = 1.333,  
xlim = c(-5.5, 5.5),  
ylim = c(0, -log10(10e-12)),  
pointSize = 1.5,  
labSize = 2.5,  
title = 'DESeq2 results',  
subtitle = 'Differential expression',  
caption = 'FC cutoff, 1.333; p-value cutoff, 10e-4',  
legendPosition = "right",  
legendLabSize = 14,  
col = c('grey30', 'forestgreen', 'royalblue', 'red2'),  
colAlpha = 0.9,  
drawConnectors = TRUE,  
hline = c(10e-8),  
widthConnectors = 0.5)
```



# Index

EnhancedVolcano, [2](#)

ggalt, [5](#)

ggplot2, [7](#)

ggrastr, [7](#)