

# Package ‘dclust’

May 8, 2026

**Type** Package

**Title** Divisive Hierarchical Clustering

**Version** 0.1.0

**Maintainer** Shaun Wilkinson <shaunwilkinson@gmail.com>

**Description** Contains a single function `dclust()` for divisive hierarchical clustering based on recursive k-means partitioning ( $k = 2$ ). Useful for clustering large datasets where computation of a  $n \times n$  distance matrix is not feasible (e.g.  $n > 10,000$  records). For further information see Steinbach, Karypis and Kumar (2000) <<http://glaros.dtc.umn.edu/gkhome/fetch/papers/docclusterKDDTMW00.pdf>>.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**URL** <http://github.com/shaunwilkinson/dclust>

**BugReports** <http://github.com/shaunwilkinson/dclust/issues>

**Imports** openssl, phylogram

**RoxygenNote** 6.1.1

**NeedsCompilation** no

**Author** Shaun Wilkinson [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-7332-7931>>),  
Paolo Giordani [aut]

**Repository** CRAN

**Date/Publication** 2019-09-05 12:00:02 UTC

## Contents

dclust	2
Index	4

---

`dclust`*Divisive/bisecting heirarchical clustering*

---

**Description**

This function recursively splits an  $n \times p$  matrix into smaller and smaller subsets, returning a "dendrogram" object.

**Usage**

```
dclust(x, method = "kmeans", stand = FALSE, ...)
```

**Arguments**

<code>x</code>	a matrix
<code>method</code>	character string giving the partitioning algorithm to be used to split the data. Currently only "kmeans" is supported (divisive/bisecting k-means; see Steinbach et al. 2000).
<code>stand</code>	logical indicating whether the matrix should be standardised prior to the recursive partitioning procedure. Defaults to FALSE.
<code>...</code>	further arguments to be passed to splitting methods (not including centers if <code>method = kmeans</code> ).

**Details**

This function creates a dendrogram by successively splitting the dataset into smaller and smaller subsets (recursive partitioning). This is a divisive, or "top-down" approach to tree-building, as opposed to agglomerative "bottom-up" methods such as neighbor joining and UPGMA. It is particularly useful for large large datasets with many records ( $n > 10,000$ ) since the need to compute a large  $n * n$  distance matrix is circumvented.

If a more accurate tree is required, users can increase the value of `nstart` passed to `kmeans` via the `...` argument. While this can increase computation time, it can improve accuracy considerably.

**Value**

Returns an object of class "dendrogram".

**Author(s)**

Shaun Wilkinson

**References**

Steinbach M, Karypis G, Kumar V (2000). A Comparison of Document Clustering Techniques. Proceedings of World Text Mining Conference, KDD2000, Boston.

## Examples

```
## Cluster a subsample of the iris dataset
suppressWarnings(RNGversion("3.5.0"))
set.seed(999)
iris50 <- iris[sample(x = 1:150, size = 50, replace = FALSE),]
x <- as.matrix(iris50[, 1:4])
rownames(x) <- iris50[, 5]
dnd <- dclust(x, nstart = 20)
plot(dnd, horiz = TRUE, yaxt = "n")

## Color labels according to species
rectify_labels <- function(node, x){
  newlab <- factor(rownames(x))[unlist(node, use.names = FALSE)]
  attr(node, "label") <- newlab
  return(node)
}
dnd <- dendrapply(dnd, rectify_labels, x = x)

## Create a color palette as a data.frame with one row for each species
uniq spp <- as.character(unique(iris50$Species))
colormap <- data.frame(Species = uniq spp, color = rainbow(n = length(uniq spp)))
colormap[, 2] <- c("red", "blue", "green")

## Color the inner dendrogram edges
color_dendro <- function(node, colormap){
  if(is.leaf(node)){
    nodecol <- colormap$color[match(attr(node, "label"), colormap$Species)]
    attr(node, "nodePar") <- list(pch = NA, lab.col = nodecol)
    attr(node, "edgePar") <- list(col = nodecol)
  }else{
    spp <- attr(node, "label")
    dominant spp <- levels(spp)[which.max(tabulate(spp))]
    edgecol <- colormap$color[match(dominant spp, colormap$Species)]
    attr(node, "edgePar") <- list(col = edgecol)
  }
  return(node)
}
dnd <- dendrapply(dnd, color_dendro, colormap = colormap)

## Plot the dendrogram
plot(dnd, horiz = TRUE, yaxt = "n")
```

# Index

dclust, 2