

# Package ‘nnclust’

January 15, 2012

**Type** Package

**Title** Nearest-neighbour tools for clustering

**Version** 2.2

**Author** Thomas Lumley

**Maintainer** Thomas Lumley <tlumley@u.washington.edu>

**Description** Finds nearest neighbours and the minimum spanning tree for large data sets, does clustering using the minimum spanning tree.

**Suggests** hexbin

**License** GPL-2

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2010-01-20 08:00:09

## R topics documented:

mst . . . . .	2
nncluster . . . . .	3
nnfind . . . . .	5
pollen . . . . .	6

<b>Index</b>	<b>7</b>
--------------	----------

---

mst *Minimum spanning trees*

---

**Description**

Minimum spanning tree by Prim's algorithm, using a fast nearest-neighbour search to find candidate points. The variant `mst_restart` stops when the link length is greater than a specified threshold.

**Usage**

```
mst(X, rebuild = sqrt(nrow(X))/4)
mst_restart(X, rebuild=sqrt(nrow(X))/4, threshold=Inf, start=NULL)
```

**Arguments**

<code>X</code>	Matrix of data points
<code>rebuild</code>	How often to rebuild the k-d tree for nearest neighbour lookup.
<code>threshold</code>	Return when the next link is longer than this
<code>start</code>	Start here rather than at the first row of <code>X</code>

**Value**

<code>from, to</code>	Row numbers of <code>X</code> for each link
<code>dist</code>	Squared length of link
<code>n</code>	For <code>mst_restart</code> only, the number of links added

**Author(s)**

Thomas Lumley

**References**

Bentley JL, Friedman JH. "Fast algorithms for constructing minimal spanning trees in coordinate spaces" IEEE Transactions on Computers C27(2) Feb 1978

**See Also**

[nncluster](#) for a wrapper for `mst_restart` to do clustering.

**Examples**

```
x<-scale(faithful)
a<-mst(x)
plot(faithful)
segments(faithful[a$from,1], faithful[a$from,2], faithful[a$to,1],
faithful[a$to,2],col="blue")
```

```
## restarting
plot(a$dist)
a<-mst_restart(x,threshold=0.04,start=0)
plot(faithful)
segments(faithful[a$from,1], faithful[a$from,2], faithful[a$to,1],
faithful[a$to,2],col="red")

b<-mst_restart(x[-a$to,],threshold=0.04)
ff<-faithful[-a$to,]
segments(ff[b$from,1], ff[b$from,2], ff[b$to,1],ff[b$to,2],col="purple")
```

---

nncluster

*Fast clustering with restarted minimum spanning tree.*


---

### Description

Uses Prim's algorithm to build a minimum spanning tree for each cluster, stopping when the nearest-neighbour distance rises above a specified threshold. Returns a set of clusters and a set of 'outliers' not in any cluster. `trimCluster` tidies up the output by removing small clusters, `clusterMember` returns cluster membership for the original data points.

### Usage

```
nncluster(x, threshold, fill = 0.95, maxclust = 20, give.up = 500, verbose=FALSE, start=NULL)
trimCluster(nnclust, size=10)
clusterMember(nnclust, outlier=TRUE)
nearestCluster(nnclust, threshold=Inf, outlier=FALSE)
```

### Arguments

<code>x</code>	data matrix
<code>threshold</code>	Threshold for stopping the tree building within a cluster. The tree stops when the squared euclidean distance to the closest point to the tree is greater than this. If <code>threshold</code> is a vector, the elements will be used in succession, with the last element repeated as necessary.
<code>fill</code>	Stop when the clusters make up this fraction of the data.
<code>maxclust</code>	Stop at this many clusters
<code>give.up</code>	Stop when fewer than this many pairs have nearest-neighbour distance less than <code>threshold</code> .
<code>verbose</code>	Print some cluster summaries before restarting?
<code>nnclust</code>	An object of class <code>nncluster</code> , returned by <code>nncluster</code>
<code>size</code>	Clusters smaller than this are added to the 'outlier' set
<code>outlier</code>	If <code>FALSE</code> , use <code>NA</code> for the cluster identifier for outliers
<code>start</code>	integer index to start the minimum spanning tree at this observation

## Details

Works best for well-separated clusters in up to 8 dimensions, and sample sizes up to hundreds of thousands.

If you want a complete minimum spanning tree, run `mst` on the outlier set and then use `nnfind` to find the shortest links connecting the clusters. When there are well-separated clusters this will be faster than running `mst` once on the whole data set.

`clusterMember` returns a vector of integers indicating cluster membership. Outliers are treated as a separate cluster if `outlier` is `TRUE`, otherwise they code as `NA`. `nearestCluster` assigns outliers at distance less than `threshold` from a cluster to the cluster whose nearest member is closest.

`trimCluster` returns a new `nncluster` object with small clusters converted to outliers. There must be at least one cluster larger than `size`.

## Value

A list of class `nncluster`. Each element but the last describes a cluster, with components `mst` containing the tree, `x` containing the data, and rows containing row numbers in the initial data set.

The last element describes the unclustered outliers and has no `mst` component.

## Note

The performance of this algorithm depends critically on the performance of the nearest-neighbour finder, and can decay catastrophically if too many uninformative variables are added.

The performance can also be poor if the data are close to being ordered on some of the variables.

## Author(s)

Thomas Lumley

## See Also

[mst](#), [nnfind](#)

## Examples

```
x<-scale(faithful)
a<-nncluster(x, threshold=0.1, give.up=0, fill=1)
a
id<-clusterMember(a)
plot(faithful, col=id, pch=19)
```

---

nnfind	<i>Nearest-neighbour distances</i>
--------	------------------------------------

---

### Description

Find the nearest neighbours of points in one data set from another data set. Useful for Mallows-type distance metrics.

### Usage

```
nnfind(from, to)
```

### Arguments

from	A matrix giving the first sample, with rows specifying points and columns specifying dimensions.
to	Optional matrix with the same number of columns as from giving the second sample.

### Details

If to is specified, for each point in to find the nearest neighbour in from. If to is not specified, for each point in from find the nearest distinct neighbour in from.

The algorithm builds a k-d tree on from and drops points down the tree to find the nearest neighbour. This is much more efficient than a brute-force search as long as the dimension is low enough. For a million points in each data set and five dimensions, the code does only about 0.03% of the possible distance computations.

### Value

count	Number of distance computations performed
neighbour	Vector of row numbers in from containing the nearest neighbour of each point in to
dist	Vector of distances to the nearest neighbour for each point in to

### Examples

```
data(faithful)
nn<-nnfind(as.matrix(faithful))
plot(faithful)
segments(faithful[,1], faithful[,2], faithful[nn$neighbour,1], faithful[nn$neighbour,2], col="blue")
```

---

pollen

*Artificial "pollen-grain" data from 1986 Data Expo.*

---

### Description

Artificial data described as measurements of pollen grains. Created for the Data Expo at the 1986 Joint Statistical Meetings by David Coleman of RCA Labs.

### Usage

```
data(pollen)
```

### Format

A data frame with 3848 observations on the following 6 variables.

ridge a numeric vector

nub a numeric vector

crack a numeric vector

weight a numeric vector

density a numeric vector

id a numeric vector

### Source

<http://lib.stat.cmu.edu/data-expo/>

### Examples

```
data(pollen)
scaled <- scale(as.matrix(pollen[,1:5]))

## clearly at least two populations, based on neighbour distances
d <- nnfind(scaled)
plot(density(d$dist))
abline(v=0.08)

## cluster to extract two populations
nn <- nncluster(scaled, threshold=c(0.08,1), give.up=0)
nn

## tight cluster is the hidden message
plot(ridge~nub, data=pollen, subset=clusterMember(nn)==1)

## data set is a hollow ellipsoid with a treat in the middle.
coplot(density~ridge|crack*nub, data=pollen, pch=".",
       col=clusterMember(nn), cex=2, n=5)
```

# Index

## \*Topic **cluster**

mst, [2](#)

nncluster, [3](#)

nnfind, [5](#)

## \*Topic **datasets**

pollen, [6](#)

clusterMember (nncluster), [3](#)

mst, [2](#), [4](#)

mst\_restart (mst), [2](#)

nearestCluster (nncluster), [3](#)

nncluster, [2](#), [3](#)

nnfind, [4](#), [5](#)

pollen, [6](#)

trimCluster (nncluster), [3](#)