# Package 'RANN'

October 12, 2022

**Title** Fast Nearest Neighbour Search (Wraps ANN Library) Using L2
Metric

**Author** Sunil Arya and David Mount (for ANN), Samuel E. Kemp, Gregory Jefferis

**Maintainer** Gregory Jefferis <jefferis@gmail.com>

**Copyright** ANN library is copyright University of Maryland and Sunil
Arya and David Mount. See file COPYRIGHT for details.

**Description** Finds the k nearest neighbours for every point in a given dataset
in O(N log N) time using Arya and Mount's ANN library (v1.1.3). There is
support for approximate as well as exact searches, fixed radius searches
and 'bd' as well as 'kd' trees. The distance is computed using the L2
(Euclidean) metric. Please see package 'RANN.L1' for the same
functionality using the L1 (Manhattan, taxicab) metric.

**URL** https://github.com/jefferis/RANN

**BugReports** https://github.com/jefferis/RANN/issues

**Encoding** UTF-8

**License** GPL (>= 3)

**Suggests** testthat

**Version** 2.6.1

**RoxygenNote** 6.1.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2019-01-08 20:00:04 UTC

## R topics documented:

---

| RANN–package | *Wrapper for Arya and Mount's Approximate Nearest Neighbours (ANN) C++ library* |
|---|---|

---

### Description

Wrapper for Arya and Mount's Approximate Nearest Neighbours (ANN) C++ library

### See Also

[nn2](nn2)

---

| nn2 | *Nearest Neighbour Search* |
|---|---|

---

### Description

Uses a kd-tree to find the p number of near neighbours for each point in an input/output dataset. The advantage of the kd-tree is that it runs in O(M log M) time.

### Usage

```
nn2(data, query = data, k = min(10, nrow(data)), treetype = c("kd",
  "bd"), searchtype = c("standard", "priority", "radius"), radius = 0,
  eps = 0)
```

### Arguments

| | |
|---|---|
| data | An **M** x **d** data.frame or matrix, where each of the **M** rows is a point or a (column) vector (where **d=1**). |
| query | A set of **N** x **d** points that will be queried against data. **d**, the number of columns, must be the same as data. If missing, defaults to data. |
| k | The maximum number of nearest neighbours to compute. The default value is set to the smaller of the number of columnns in data |
| treetype | Character vector specifying the standard 'kd' tree or a 'bd' (box-decomposition, AMNSW98) tree which may perform better for larger point sets |
| searchtype | See details |
| radius | Radius of search for searchtype='radius' |
| eps | Error bound: default of 0.0 implies exact nearest neighbour search |

## Details

The RANN package utilizes the Approximate Near Neighbor (ANN) C++ library, which can give the exact near neighbours or (as the name suggests) approximate near neighbours to within a specified error bound. For more information on the ANN library please visit [http://www.cs.umd.edu/~mount/ANN/](http://www.cs.umd.edu/~mount/ANN/).

Search types: `priority` visits cells in increasing order of distance from the query point, and hence, should converge more rapidly on the true nearest neighbour, but standard is usually faster for exact searches. `radius` only searches for neighbours within a specified radius of the point. If there are no neighbours then nn.idx will contain 0 and nn.dists will contain 1.340781e+154 for that point.

## Value

A `list` of length 2 with elements:

| | |
|---|---|
| nn.idx | A **N** x **k** integer `matrix` returning the near neighbour indices. |
| nn.dists | A **N** x **k** `matrix` returning the near neighbour Euclidean distances. |

## Author(s)

Gregory Jefferis based on earlier code by Samuel E. Kemp (knnFinder package)

## References

Bentley J. L. (1975), Multidimensional binary search trees used for associative search. Communication ACM, 18:309-517.

Arya S. and Mount D. M. (1993), Approximate nearest neighbor searching, Proc. 4th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'93), 271-280.

Arya S., Mount D. M., Netanyahu N. S., Silverman R. and Wu A. Y (1998), An optimal algorithm for approximate nearest neighbor searching, Journal of the ACM, 45, 891-923.

## Examples

```
x1 <- runif(100, 0, 2*pi)
x2 <- runif(100, 0,3)
DATA <- data.frame(x1, x2)
nearest <- nn2(DATA,DATA)
```

# Index