# Package ‘intrinsicDimension’

October 13, 2022

**Type** Package

**Title** Intrinsic Dimension Estimation

**Version** 1.2.0

**Date** 2019-05-23

**Author** Kerstin Johnsson, Lund University

**Maintainer** Kerstin Johnsson <kerstin.johnsson@hotmail.com>

**Depends** yaImpute

**Description** A variety of methods for estimating intrinsic dimension of data sets (i.e the manifold or Hausdorff dimension of the support of the distribution that generated the data) as reviewed in Johnsson, K. (2016, ISBN:978-91-7623-921-6) and Johnsson, K., Soneson, C. and Fontes, M. (2015) <doi:10.1109/TPAMI.2014.2343220>. Furthermore, to evaluate the performance of these estimators, functions for generating data sets with given intrinsic dimensions are provided.



**LazyLoad** yes

**RoxygenNote** 6.0.1

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2019-06-07 10:20:03 UTC

## R topics documented:

---

`intrinsicDimension-package` *Intrinsic Dimension Estimation and Data on Manifolds*

---

## Description

The intrinsic dimension of a data set is a measure of its complexity. In technical terms it typically means the manifold or Hausdorff (fractal) dimension of the support of the probability distribution generating the data. This package contains functions for estimating intrinsic dimension and generating ground truth data sets with known intrinsic dimension.

## Details

Data sets that can be accurately described with a few parameters have low intrinsic dimension. It is expected that the performance of many machine learning algorithms is dependent on the intrinsic dimension of the data. Is has also been proposed to use estimates of intrinsic dimension for applications such as network anomaly detection and image analysis.

This package contains implementations of a variety of approaches for intrinsic dimension estimation: modeling distances by for example Maximum Likelihood, approximating hyperplanes using Principal Component Analysis (PCA) and modeling angular information and concentration of measure (ESS and DANCo methods). Ground truth data, i.e. data with known intrinsic dimension, can be generated with a number of functions modeling manifolds. The manifold dimension is the intrinsic dimension.

The package distinguishes between local, global and pointwise estimators of intrinsic dimension. Local estimators estimate dimension of a _local data set_, for example a neighborhood from a larger data set. For this estimate to be accurate the noise and the curvature of the data has to be small relative to the neighborhood diameter. A global estimator takes the entire data set and returns one estimate of intrinsic dimension. Global estimators has the potential to handle higher noise and curvature levels than local estimators, but require that the entire data set has the same intrinsic dimension. Pointwise estimators are essentially local estimators applied neighborhoods around each point in the data set, but sometimes information beyond the neighborhood is used, as in PCA with Optimally Topology Preserving Maps. Any local estimator can be converted into a pointwise estimator.

Functions for estimating intrinsic dimension: `localIntrinsicDimension`, `globalIntrinsicDimension`, `pointwiseIntrinsicDimension`, `essLocalDimEst`, `dancoDimEst`, `pcaLocalDimEst`, `pcaOtpmPointwiseDimEst`, `maxLikGlobalDimEst`, `maxLikLocalDimEst`, `maxLikPointwiseDimEst`, `knnDimEst`.

Functions for generating data points from (usually uniform) distributions on manifolds (possibly with noise): `hyperBall`, `hyperSphere`, `hyperCube`, `isotropicNormal`, `hyperCubeFaces`, `hyperCubeEdges`, `cutHyperPlane`, `cutHyperSphere`, `oblongNormal`, `swissRoll`, `swissRoll3Sph`, `twinPeaks`, `hyperTwinPeaks`, `cornerPlane`, `mHeinManifold`, `m14Manifold`, `m15Manifold`.

Functions for applying local estimators to non-local data: `asPointwiseEstimator`, `neighborhoods`

## Author(s)

Kerstin Johnsson, Lund University

Maintainer: Kerstin Johnsson <kerstin.johnsson@hotmail.com>

## References

Johnsson, K (2016). Structures in high-dimensional data: Intrinsic dimension and cluster analysis. PhD thesis. Lund University. http://portal.research.lu.se/portal/sv/publications/structures-in-highdimensional-data-intrinsic-dimension-and-cluster-analysis(8404f72e-e760-436d-ad7
.html

---

`addNoise` *Add Noise to Data Set*

---

## Description

Embeds the data in `n` dimensions and adds normal isotropic noise to the data set. Hence `n` has to be at least equal to the dimension (the number of columns) of the data set, otherwise the function terminates with an error.

## Usage

```
addNoise(data, n = ncol(data), sd)
```

## Arguments

| | |
|---|---|
| `data` | data set. Each row corresponds to a data point. |
| `n` | dimension of noise. |
| `sd` | standard deviation of noise. The covariance matrix of the noise is $sd^2 \cdot I$. |

## Value

Matrix of same size as `data`.

## Author(s)

Kerstin Johnsson, Lund University

## Examples

```
datap <- hyperCubeEdges(100, 1, 2)
datap <- addNoise(datap, 3, .05)
par(mfrow = c(1, 2))
plot(datap[, 1], datap[, 2])
plot(datap[, 1], datap[, 3])
```

---

# `asPointwiseEstimator` *Turn a local estimator into a pointwise estimator*

---

## Description

Returns a function that can be used as a pointwise estimator of intrinsic dimension that uses local data sets with a fixed number of data points.

## Usage

```
asPointwiseEstimator(estimator, neighborhood.size, indices=NULL, eps=0.0)
```

## Arguments

| | |
|---|---|
| `estimator` | A local intrinsic dimension estimator. |
| `neighborhood.size` | The number of neighbors used for each dimension estimate. |
| `indices` | A vector with indices of the points in `data` (as sent to the estimator function) that should be used as center for neighborhoods. |
| `eps` | If non-zero, the relative error in distance allowed when finding nearest neighbors. See Details. |

## Details

The `ann` function of the package `yaImpute` is used for finding the `k` nearest neighbors. The `eps` parameter to `neighborhoods` is used in the `ann` function.

## Value

A function that can be used as a pointwise dimension estimator.

### Author(s)

Kerstin Johnsson, Lund University

### Examples

```
data <- swissRoll3Sph(300, 300)
 # the first 300 data points are on the swiss roll (ID=2) , the last 300 on the 3-sphere (ID=3)
essPointwiseDimEst <- asPointwiseEstimator(essLocalDimEst, neighborhood.size=10,
                                           indices = c(1:10, 301:310))
ess.pw.res <- essPointwiseDimEst(data)
ess.pw.res$dim.est
```

---

## cornerPlane *Corner Plane*

---

### Description

Generates a sample from a uniform distribution on a bent plane. Half of the plane is in the xz-plane and half of the plane is bent over the x-axis, so that the resulting surface has an edge along the x-axis.

### Usage

```
cornerPlane(Ns, theta = pi/4)
```

### Arguments

| | |
|---|---|
| `Ns` | number of data points. |
| `theta` | angle at the x-axis. |

### Value

A `Ns` x 3 matrix with columns x, y and z.

### Author(s)

Kerstin Johnsson, Lund University

### Examples

```
datap <- cornerPlane(400)
par(mfrow = c(1, 2))
plot(datap[,1], datap[,2])
plot(datap[,1], datap[,3])
```

cutHyperPlane *Piece of Noisy Hyperplane*

## Description

Generates Ns data points within the unit ball from a hyperplane through the origin with noise added. n has to be at least d, otherwise the function terminates with an error.

## Usage

```
cutHyperPlane(Ns, d, n, sd)
```

## Arguments

| | |
|---|---|
| Ns | number of data points. |
| d | dimension of hyperplane. |
| n | dimension of noise. |
| sd | standard deviation of noise. |

## Details

The data set is generated the following way: First data points are sampled uniformly in a d-ball. After this, (n-d)-dimensional orthogonal noise with standard deviation sd in each direction is added. No noise is added in the directions parallel to the hyperplane since on an infinite plane adding isotropic noise to a uniform distribution does not change the distribution. Finally all data points within distance 1 from the origin are considered as candidates for the data set that will be returned, out of the candidates Ns data points are chosen randomly to be returned. If there are less than Ns candidates more candidates will be generated in the same way.

The data generated by this function can be used to evaluate how much local dimension estimators are affected by noise.

## Value

A Ns x n matrix.

## Warning

If sd is high, cutHyperPlane will be slow and might not even be able to return a data set. If so, it will return NULL.

## Author(s)

Kerstin Johnsson, Lund University

## See Also

cutHyperSphere

## Examples

```
datap <- cutHyperPlane(100, 2, 3, 0.01)
par(mfrow = c(1, 2))
plot(datap[, 1], datap[, 2])
plot(datap[, 1], datap[, 3])
```

---

`cutHyperSphere` *Piece of Noisy Hypersphere*

---

## Description

Generates `Ns` data points cut out from a noisy hypersphere. `n` has to be at least `d+1`, otherwise the function terminates with an error.

## Usage

```
cutHyperSphere(Ns, rat, d, n, sd)
```

## Arguments

| | |
|---|---|
| `Ns` | number of data points. |
| `rat` | ratio between cut-off radius and radius of sphere. |
| `d` | (intrinsic) dimension of hypersphere. |
| `n` | dimension of noise. |
| `sd` | standard deviation of noise. |

## Details

The returned data are within distance `rat` the point $1/\sqrt{d+1}(1...1)$ and are obtained from a unit distribution on the `d`-sphere overlaid with `n`-dimensional normal noise.

The data generated by this function can be used to evaluate the performance of local dimension estimators.

## Value

A `Ns` by `n` matrix.

## Warning

If `sd` is high, `cutHyperSphere` will be slow and might not even be able to return a data set. If so, it will return `NULL`.

## Author(s)

Kerstin Johnsson, Lund University

## See Also

cutHyperPlane

## Examples

```
datap <- cutHyperSphere(100, rat = .5, 1, 3, 0.01)
par(mfrow = c(1, 2))
plot(datap[, 1], datap[, 2])
plot(datap[, 1], datap[, 3])

datap <- cutHyperSphere(100, rat = 2, 1, 3, 0.11)
par(mfrow = c(1, 2))
plot(datap[, 1], datap[, 2])
plot(datap[, 1], datap[, 3])
```

---

`dancoDimEst` *Dimension Estimation With the DANCo and MIND Methods*

---

## Description

Intrinsic dimension estimation with the DANCo (Ceruti et al. 2012), MIND_MLi and MIND_MLk (Rozza et al. 2012) methods.

## Usage

```
dancoDimEst(data, k, D, ver = "DANCo", calibration.data = NULL)
```

## Arguments

| | |
|---|---|
| `data` | a data set for which the intrinsic dimension is estimated. |
| `k` | neighborhood parameter. |
| `D` | maximal dimension. |
| `ver` | possible values: 'DANCo', 'MIND_MLi', 'MIND_MLk'. |
| `calibration.data` | precomputed calibration data. |

## Details

If `cal` = `NULL` or the `cal$maxdim` < `D` new calibration data will be computed as needed.

## Value

A `DimEst` object with slots:

| | |
|---|---|
| `dim.est` | the intrinsic dimension estimate. |
| `kl.divergence` | the KL divergence between data and reference data for the estimated dimension (if ver == 'DANCo'). |

calibration.data
: calibration data that can be reused when applying DANCo to data sets of the same size with the same neighborhood parameter k.

## Author(s)

Kerstin Johnsson, Lund University

## References

Ceruti, C. et al. (2012) DANCo: Dimensionality from Angle and Norm Concentration. *arXiv preprint* 1206.3881.

Rozza, A et al. (2012) Novel high intrinsic dimensionality estimators. *Machine learning* **89**, 37-65.

## Examples

```
data <- hyperBall(50, 10)
res <- dancoDimEst(data, 8, 20)
print(res)

## Reusing calibration data
data2 <- hyperBall(50, 5)
dancoDimEst(data2, 8, 20, calibration.data=res$calibration.data)
```

---

# essLocalDimEst — *Expected Simplex Skewness Local Dimension Estimation*

---

## Description

Local intrinsic dimension estimation with the ESS method

## Usage

```
essLocalDimEst(data, ver, d = 1)
```

## Arguments

| | |
|---|---|
| data | Local data set for which dimension should be estimated. |
| ver | Possible values: 'a' and 'b'. See Johnsson et al. (2015). |
| d | For ver = 'a', any value of d is possible, for ver = 'b', only d = 1 is supported. |

## Details

The ESS method assumes that the data is local, i.e. that it is a neighborhood taken from a larger data set, such that the curvature and the noise within the neighborhood is relatively small. In the ideal case (no noise, no curvature) this is equivalent to the data being uniformly distributed over a hyper ball.

### Value

A `DimEst` object with two slots:

| | |
|---|---|
| `dim.est` | The interpolated dimension estimate. |
| `ess` | The ESS value produced by the algorithm. |

### Author(s)

Kerstin Johnsson, Lund University

### References

Johnsson, K., Soneson, C., & Fontes, M. (2015). Low Bias Local Intrinsic Dimension Estimation from Expected Simplex Skewness. IEEE Trans. Pattern Anal. Mach. Intell., 37(1), 196-202.

### Examples

```
data <- hyperBall(100, 4, 15, .05)
essLocalDimEst(data, ver = 'a', d = 1)
```

---

## `essReference` *ESS Reference Values*

---

### Description

Reference values for the ESS dimension estimation method

### Usage

```
essReference(ver, d, maxdim, mindim=1)
```

### Arguments

| | |
|---|---|
| `ver` | Possible values: `'a'` and `'b'`. See Johnsson et al. (2015). |
| `d` | For `ver = 'a'`, any value of `d` is possible, for `ver = 'b'`, only `d = 1` is supported. |
| `maxdim` | The largest dimension for which reference values should be computed. |
| `mindim` | The smallest dimension for which reference values should be computed. |

### Details

The ESS reference values are used by the ESS algorithm (essLocalDimEst) to compute the final dimension estimate.

### Value

A vector of length maxdim-(mindim-1), where each slot represents the reference value.

### Author(s)

Kerstin Johnsson, Lund University

### References

Johnsson, K., Soneson, C., & Fontes, M. (2015). Low Bias Local Intrinsic Dimension Estimation from Expected Simplex Skewness. IEEE Trans. Pattern Anal. Mach. Intell., 37(1), 196-202.

### Examples

```
essReference('a', 3, maxdim=500)
essReference('b', 1, maxdim=30, mindim=3)
```

---

## `hyperCube` *Hypercube*

---

### Description

Generates a sample from a uniform distribution on a hypercube, the faces of a hypercube or the "edges" of a hyper cube.

### Usage

```
hyperCube(Ns, n, side = 1)
hyperCubeFaces(Ns, n)
hyperCubeEdges(Ns, d, n)
```

### Arguments

| | |
|---|---|
| `Ns` | number of data points. |
| `d` | dimension of edges. |
| `n` | dimension of the hypercube. |
| `side` | the length of the side of the hyper cube. |

### Details

The hypercube is $[0, 1]^n$. The edges of dimension `d` of the hypercube are the `d`-dimensional boundaries of the hypercube. The hypercube faces are the hyper cube edges of dimension `n-1`.

### Value

A `Ns` by `n` matrix.

### Author(s)

Kerstin Johnsson, Lund University.

## Examples

```
datap <- hyperCubeEdges(200, 1, 3)
par(mfrow = c(1, 3))
plot(datap[, 1], datap[, 2])
plot(datap[, 1], datap[, 3])
plot(datap[, 2], datap[, 3])
```

---

`ide` *Intrinsic Dimension Estimation*

---

## Description

Intrinsic dimension estimation with method given as parameter.

## Usage

```
localIntrinsicDimension(.data, .method, ...)
globalIntrinsicDimension(.data, .method, ...)
pointwiseIntrinsicDimension(.data, .method, ...)
```

## Arguments

| | |
|---|---|
| `.data` | Data set for which dimension should be estimated. |
| `.method` | For `local.dimension.estimation`, one of `'essLocalDimEst'`, `'dancoDimEst'`, `'pcaLocalDimEst'`, `'maxLikLocalDimEst'`, `'knnDimEst'`. For `global.dimension.estimation`, one of `'dancoDimEst'`, `'maxLikGlobalDimEst'`, `'knnDimEst'`. For `pointwise.dimension.estimatio` `'pcaOtpmLocalDimEst'` or `'maxLikPointwiseDimEst'`. |
| `...` | arguments passed to intrinsic dimension estimator. |

## Details

For the `localIntrinsicDimension` function, `.data` should be a local data set, i.e. a piece of a data set that is well approximated by a hyperplane (meaning that the curvature should be low in the local data set).

The function `pointwiseIntrinsicDimension` estimates local dimension around each data point in the data set.

## Value

For `localIntrinsicDimension` and `globalIntrinsicDimension`, a `DimEst` object with the slot `dim.est` containing the dimension estimate and possibly additional slots containing additional information about the estimation process.

For `pointwiseIntrinsicDimension`, a `DimEstPointwise` object, inheriting `data.frame`, with one slot `dim.est` containing the dimension estimates and possibly additional slots containing additional information about the estimation process.

## Author(s)

Kerstin Johnsson, Lund University

## References


Johnsson, K (2016). Structures in high-dimensional data: Intrinsic dimension and cluster analysis. PhD thesis. Lund University.

Johnsson, K., Soneson, C. and Fontes, M. (2015). Low Bias Local Intrinsic Dimension Estimation from Expected Simplex Skewness. *IEEE Trans. Pattern Anal. Mach. Intell.*, **37**(1), 196-202.

Ceruti, C. et al. (2012). DANCo: Dimensionality from Angle and Norm Concentration. *arXiv preprint* 1206.3881.

Rozza, A et al. (2012). Novel high intrinsic dimensionality estimators. *Machine learning* **89**, 37-65.

Fukunaga, K. and Olsen, D. R. (1971). An algorithm for finding intrinsic dimensionality of data. *IEEE Trans. Comput.*, **c-20**(2):176-183.

Fan, M. et al. (2010). Intrinsic dimension estimation of data by principal component analysis. *arXiv preprint* 1002.2050.

Bruske, J. and Sommer, G. (1998) Intrinsic dimensionality estimation with optimally topology preserving maps. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, **20**(5), 572-575.

Haro, G., Randall, G. and Sapiro, G. (2008) Translated Poisson Mixture Model for Stratification Learning. *Int. J. Comput. Vis.*, **80**, 358-374.

Hill, B. M. (1975) A simple general approach to inference about the tail of a distribution. *Ann. Stat.*, **3**(5) 1163-1174.

Levina, E. and Bickel., P. J. (2005) Maximum likelihood estimation of intrinsic dimension. *Advances in Neural Information Processing Systems 17*, 777-784. MIT Press.

Carter, K.M., Raich, R. and Hero, A.O. (2010) On local intrinsic dimension estimation and its applications. *IEEE Trans. on Sig. Proc.*, **58**(2), 650-663.


## See Also

`essLocalDimEst`, `dancoDimEst`, `pcaLocalDimEst`, `knnDimEst` `pcaOtpmPointwiseDimEst`, `maxLikGlobalDimEst`, `maxLikLocalDimEst`, `maxLikPointwiseDimEst`

## Examples

```
data <- hyperBall(100, 4, 15, .05)
localIntrinsicDimension(data, .method='essLocalDimEst', ver = 'a', d = 1)
globalIntrinsicDimension(data, 'dancoDimEst', k = 8, D = 20)
pointwiseIntrinsicDimension(data, .method='maxLikPointwiseDimEst', k = 8, dnoise = NULL)
```

`knnDimEst` *Dimension Estimation from kNN Distances*

## Description

Estimates the intrinsic dimension of a data set using weighted average kNN distances.

## Usage

```
knnDimEst(data, k, ps, M, gamma = 2)
```

## Arguments

| | |
|---|---|
| `data` | data set with each row describing a data point. |
| `k` | number of distances to neighbors used at a time. |
| `ps` | vector with sample sizes; each sample size has to be larger than k and smaller than `nrow(data)`. |
| `M` | number of bootstrap samples for each sample size. |
| `gamma` | weighting constant. |

## Details

This is a somewhat simplified version of the kNN dimension estimation method described by Carter et al. (2010), the difference being that block bootstrapping is not used.

## Value

A `DimEst` object with slots:

| | |
|---|---|
| `dim.est` | the intrinsic dimension estimate (integer). |
| `residual` | the residual, see Carter et al. (2010). |

## Author(s)

Kerstin Johnsson, Lund University.

## References

Carter, K.M., Raich, R. and Hero, A.O. (2010) On local intrinsic dimension estimation and its applications. *IEEE Trans. on Sig. Proc.*, **58**(2), 650-663.

## Examples

```
N <- 50
data <- hyperBall(N, 5)

k <- 2
ps <- seq(max(k + 1, round(N/2)), N - 1, by = 3)
knnDimEst(data, k, ps, M = 10, gamma = 2)
```

---

# mHeinManifold *12-dimensional manifold from Hein and Audibert (2005)*

---

## Description

Generates a 12-dimensional manifold with extrinsic dimension 72 (not uniformly sampled).

## Usage

```
mHeinManifold(Ns)
```

## Arguments

| | |
|---|---|
| Ns | number of data points. |

## Value

A 72-dimensional data set.

## Author(s)

Kerstin Johnsson, Lund University.

## References

Hein, M. and Audibert, J-Y. (2005) Intrinsic Dimensionality Estimation of Submanifolds in R^d. *Proceedings of ICML*, 289-296.

## Examples

```
datap <- mHeinManifold(800)
par(mfrow = c(1, 3))
plot(datap[,1], datap[,3])
plot(datap[,2], datap[,3])
plot(datap[,1], datap[,2])
```

M_rozza *Manifolds from Rozza et al. (2012)*

## Description

Generates data sets from Rozza et al. (2012). M14 is an 18-dimensional manifold with intrinsic dimension 72. M14 is a 24-dimensional manifold with extrinsic dimension 96. Note that M14 and M15 are not uniformly sampled.

## Usage

```
m14Manifold(Ns)
m15Manifold(Ns)
```

## Arguments

| | |
|---|---|
| `Ns` | number of data points. |

## Value

A `72`-dimensional or `96`-dimensional data set respectively.

## Author(s)

Kerstin Johnsson, Lund University.

## References

Rozza, A. et al. (2012) Novel high intrinsic dimensionality estimators. *Machine Learning*, 89:37-65.

## Examples

```
datap <- m14Manifold(800)
par(mfrow = c(1, 3))
plot(datap[,1], datap[,3])
plot(datap[,2], datap[,3])
plot(datap[,1], datap[,2])
datap <- m15Manifold(800)
par(mfrow = c(1, 3))
plot(datap[,1], datap[,3])
plot(datap[,2], datap[,3])
plot(datap[,1], datap[,2])
```

`neighborhoods` *Obtaining neighborhoods (local data) from a data set*

## Description

Get a list of neighborhoods, each containing the `k` nearest neighbors (not including itself) to a point in the data set.

## Usage

```
neighborhoods(data, k, indices, eps=0.0)
```

## Arguments

| | |
|---|---|
| `data` | A data set. |
| `k` | The number of neighbors in each neighborhood. |
| `indices` | A vector with indices of the points in `data` that should be used as center for neighborhoods. |
| `eps` | If non-zero, the relative error in distance allowed when finding nearest neighbors. See Details. |

## Details

The `ann` function of the package `yaImpute` is used for finding the `k` nearest neighbors. The `eps` parameter to `neighborhoods` is used in the `ann` function.

## Value

A list of neighborhoods where each item corresponds to one index in `indices` and each item contains a data set with `k` data points.

## Author(s)

Kerstin Johnsson, Lund University

## Examples

```
data <- swissRoll3Sph(300, 300)
neighborhoods(data, 10, 1:10)
```

Noisefun *Transition Functions Describing Noise*

## Description

Transition functions $f(s|r)$ describing the shift in lengths of vectors when Gaussian noise is added. Given a length $r$, $f(s|r)$ is the probability density for the length after noise is added to one endpoint.

## Usage

```
dnoiseNcChi(r, s, sigma, k)
dnoiseGaussH(r, s, sigma, k)
```

## Arguments

| | |
|---|---|
| r | length or vector of lengths of original vector. |
| s | length or vector of lengths of perturbed vector. |
| sigma | noise standard deviation. |
| k | dimension of noise. |

## Details

`dnoiseNcChi` is the true transition function density when the noise is Gaussian, the other transition functions are approximations of this. `dnoiseGaussH` is the Gaussian approximation used in Haro et al.

If Gaussian noise is added to both endpoints of the vector, `sigma` should be replaced by `sqrt(2)*sigma`.

## Value

Vector of probability densities.

## Note

Only `r` or `s` can be a vector.

## Author(s)

Kerstin Johnsson, Lund University

## References

Haro, G., Randall, G. and Sapiro, G. (2008) Translated Poisson Mixture Model for Stratification Learning. *Int. J. Comput. Vis.*, **80**, 358-374.

## See Also

`maxLikPointwiseDimEst`, `maxLikGlobalDimEst`, `maxLikLocalDimEst`

## Examples

```
# High SNR, high-dim noise
sigma <- 0.05
x <- seq(0, 1.5, length.out = 200)
y <- dnoiseNcChi(x, s = .5, sigma, k = 20)
plot(x, y, type = 'l', main = 'Noise dim = 20')
y2 <- dnoiseGaussH(x, s = .5, sigma, k = 20)
lines(x, y2, lty = 2)

# Low SNR
par(mfrow = c(2, 3))
sigma <- 0.2
x <- seq(0, 1.5, length.out = 200)
y <- dnoiseNcChi(x, s = .5, sigma, k = 4)
plot(x, y, type = 'l', main = 'Noise approximations')
y2 <- dnoiseGaussH(x, s = .5, sigma, k)
lines(x, y2, lty = 2)

# High SNR, low-dim noise
sigma <- 0.05
x <- seq(0, 1.5, length.out = 200)
y <- dnoiseNcChi(x, s = .5, sigma, k = 4)
plot(x, y, type = 'l', main = 'Noise dim = 4')
y2 <- dnoiseGaussH(x, s = .5, sigma, k)
lines(x, y2, lty = 2)
```

---

`oblongNormal` *Oblong Normal Distribution*

---

## Description

Generates a sample from a certain anisotropic normal distribution centered around the origin.

## Usage

```
oblongNormal(Ns, n)
```

## Arguments

| | |
|---|---|
| `Ns` | number of data points. |
| `n` | dimension of the distribution (and the data points). |

## Details

In the first half of the dimensions (rounded down if `n` is odd) the standard deviation is 1 and in the rest the standard deviation is 0.25 .

### Value

A Ns by n matrix.

### Author(s)

Kerstin Johnsson, Lund University

### Examples

```
datap <- oblongNormal(100, 10)
par(mfrow = c(1, 2))
plot(datap[, 1], datap[, 2])
plot(datap[, 1], datap[, 6])
```

---

## pcaLocalDimEst *Local Dimension Estimation with PCA*

---

### Description

Estimates local manifold dimension using the largest singular values of the covariance matrix.

### Usage

```
pcaLocalDimEst(data, ver, alphaFO = .05, alphaFan = 10, betaFan = .8, PFan = .95,
     ngap = 5, maxdim = min(dim(data)), verbose = TRUE)
```

### Arguments

| | |
|---|---|
| data | a local data set for which dimension should be estimated. |
| ver | possible values: 'FO', 'fan', 'maxgap', 'cal'. 'cal' is often very slow. |
| alphaFO | only for ver = 'FO'. An eigenvalue is considered significant if it is larger than alpha times the largest eigenvalue. |
| alphaFan | only for ver = 'Fan'. The alpha parameter (large gap threshold). |
| betaFan | only for ver = 'Fan'. The beta parameter (total covariance threshold). |
| PFan | only for ver = 'Fan'. Total covariance in non-noise. |
| ngap | only for ver = 'cal'. How many of the largest gaps that should be considered. |
| maxdim | only for ver = 'cal'. The maximal manifold dimension of the data. |
| verbose | should information about the process be printed out? |

### Details

Version 'FO' is the method by Fukunaga-Olsen, version 'fan' is the method by Fan et al..

Version 'maxgap' returns the position of the largest relative gap in the sequence of singular values.

Version 'cal' considers the positions of the ngap largest relative gaps in the sequence of singular values and generates calibration data to determine which one of them is most likely.

All versions assume that the data is local, i.e. that it is a neighborhood taken from a larger data set, such that the curvature and the noise within the neighborhood is relatively small. In the ideal case (no noise, no curvature) this is equivalent to the data being uniformly distributed over a hyper ball.

### Value

A DimEst object with slots:

| | |
|---|---|
| dim.est | the dimension estimate |
| gap.size | if ver is not 'cal', the size of the gap in singular values corresponding to the estimated dimension |
| likelihood | if ver is cal, the likelihood of the estimated dimension. |

### Author(s)

Kerstin Johnsson, Lund University

### References

Fukunaga, K. and Olsen, D. R. (1971). An algorithm for finding intrinsic dimensionality of data. *IEEE Trans. Comput.*, **c-20**(2):176-183.

Fan, M. et al. (2010). Intrinsic dimension estimation of data by principal component analysis. *arXiv preprint* 1002.2050.

### See Also

pcaOtpmPointwiseDimEst

### Examples

```
data <- cutHyperPlane(100, 4, 10, .05)
pcaLocalDimEst(data, 'fan')
pcaLocalDimEst(data, 'FO')
pcaLocalDimEst(data, 'maxgap')
```

## pcaOtpmPointwiseDimEst

*Dimension Estimation With Optimally Topology Preserving Maps*

### Description

Intrinsic dimension estimation with the method proposed in Bruske and Sommer (1998). A graph called optimally topology preserving map (OTPM) is constructed and on this local PCA is made with the Fukunaga-Olsen criterion to determine which eigenvalues that are significant.

### Usage

```
pcaOtpmPointwiseDimEst(data, N, alpha = .05)
```

### Arguments

| | |
|---|---|
| `data` | a data set for which dimension should be estimated. |
| `N` | the number of the nodes in the OTPM. |
| `alpha` | the significance level for the Fukunaga-Olsen method. |

### Value

A `DimEstPointwise` object, inheriting `data.frame`, with two columns:

| | |
|---|---|
| `dim.est` | The dimension estimate at each point. |
| `nbr.nb` | The number of neighboring nodes used for the dimension estimate at each point. |

### Author(s)

Kerstin Johnsson, Lund University

### References

Bruske, J. and Sommer, G. (1998) Intrinsic dimensionality estimation with optimally topology preserving maps. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, **20**(5), 572-575.

### See Also

`pcaLocalDimEst`

### Examples

```
data <- hyperBall(1000, 5)
pcaOtpmPointwiseDimEst(data, 400)
```

`Spherical` *Isotropic Distributions With or Without Noise*

## Description

Generates a sample from isotropic distributions in `d` dimensions with `n`-dimensional noise added to it.

## Usage

```
hyperBall(Ns, d, n = d, sd = 0)
hyperSphere(Ns, d, n = d + 1, sd = 0)
isotropicNormal(Ns, d, n = d, sd = 0)
```

## Arguments

| | |
|---|---|
| `Ns` | number of points. |
| `d` | intrinsic dimension of the support of the distribution (the manifold.) |
| `n` | dimension of noise. |
| `sd` | standard deviation of noise. |

## Details

`hyperBall` draws a sample from a uniform distribution on a hyper ball of radius 1. `hyperSphere` draws a sample from a uniform distribution on a hypersphere of radius 1. `isotropicNormal` draws a sample from a isotropic normal distribution with identity covariance matrix.

## Author(s)

Kerstin Johnsson, Lund University

## Examples

```
datap <- hyperSphere(100, 1, 3, sd = .1)
par(mfrow = c(1, 2))
plot(datap[, 1], datap[, 2])
plot(datap[, 1], datap[, 3])
```

swissRoll *Swiss roll with or without 3-sphere inside*

## Description

Generates a sample from a uniform distribution on a Swiss roll-surface, possibly together with a sample from a uniform distribution on a 3-sphere inside the Swiss roll.

## Usage

```
swissRoll(Ns, a = 1, b = 2, nturn = 1.5, h = 4)
swissRoll3Sph(Ns, Nsph, a = 1, b = 2, nturn = 1.5, h = 4)
```

## Arguments

| | |
|---|---|
| Ns | number of data points on the Swiss roll. |
| Nsph | number of data points on the 3-sphere. |
| a | minimal radius of Swiss roll and radius of 3-sphere. |
| b | maximal radius of Swiss roll. |
| nturn | number of turns of the surface. |
| h | height of Swiss roll. |

## Value

`swissRoll` returns three-dimensional data points. `swissRoll3Sph` returns four-dimensional data points with the Swiss roll in the three first dimensions (columns). The `Ns` first data points lie on the Swiss roll and the `Nsph` last data points lie on the 3-sphere.

## Author(s)

Kerstin Johnsson, Lund University.

## Examples

```
datap <- swissRoll3Sph(300, 100)
par(mfrow = c(1, 3))
plot(datap[,1], datap[,2])
plot(datap[,1], datap[,3])
plot(datap[,1], datap[,4])
```

tp *Dimension Estimation via Translated Poisson Distributions*

## Description

Estimates the intrinsic dimension of a data set using models of translated Poisson distributions.

## Usage

```
maxLikGlobalDimEst(data, k, dnoise = NULL, sigma = 0, n = NULL,
        integral.approximation = 'Haro', unbiased = FALSE,
        neighborhood.based = TRUE,
        neighborhood.aggregation = 'maximum.likelihood', iterations = 5, K = 5)
maxLikPointwiseDimEst(data, k, dnoise = NULL, sigma = 0, n = NULL, indices = NULL,
             integral.approximation = 'Haro', unbiased = FALSE, iterations = 5)
maxLikLocalDimEst(data, dnoise = NULL, sigma = 0, n = NULL,
       integral.approximation = 'Haro',
       unbiased = FALSE, iterations = 5)
```

## Arguments

| | |
|---|---|
| `data` | data set with each row describing a data point. |
| `k` | the number of distances that should be used for each dimension estimation. |
| `dnoise` | a function or a name of a function giving the translation density. If NULL, no noise is modeled, and the estimator turns into the Hill estimator (see References). Translation densities `dnoiseGaussH` and `dnoiseNcChi` are provided in the package. `dnoiseGaussH` is an approximation of `dnoiseNcChi`, but faster. |
| `sigma` | (estimated) standard deviation of the (isotropic) noise. |
| `n` | dimension of the noise. |
| `indices` | the indices of the data points for which local dimension estimation should be made. |
| `integral.approximation` | how to approximate the integrals in eq. (5) in Haro et al. (2008). Possible values: `'Haro'`, `'guaranteed.convergence'`, `'iteration'`. See Details. |
| `unbiased` | if `TRUE`, a factor `k-2` is used instead of the factor `k-1` that was used in Haro et al. (2008). This makes the estimator is unbiased in the case of data without noise or boundary. |
| `neighborhood.based` | if TRUE, dimension estimation is first made for neighborhoods around each data point and final value is aggregated from this. Otherwise dimension estimation is made once, based on distances in entire data set. |
| `neighborhood.aggregation` | if `neighborhood.based`, how should dimension estimates from different neighborhoods be combined. Possible values: `'maximum.liklihood'` follows Haro |

et al. (2008) in maximizing likelihood by using the harmonic mean, `'mean'` follows Levina and Bickel (2005) and takes the mean, `'robust'` takes the median, to remove influence from possible outliers.

`iterations` for `integral.approxmation = 'iteration'`, how many iterations should be made.

`K` for `neighborhood.based = FALSE`, how many distances for each data point should be considered when looking for the `k` shortest distances in the entire data set.

## Details

The estimators are based on the referenced paper by Haro et al. (2008), using the assumption that there is a single manifold. The estimator in the paper is obtained using default parameters and `dnoise = dnoiseGaussH`.

With `integral.approximation = 'Haro'` the Taylor expansion approximation of `r^(m-1)` that Haro et al. (2008) used are employed. With `integral.approximation = 'guaranteed.convergence'`, `r` is factored out and kept and `r^(m-2)` is approximated with the corresponding Taylor expansion. This guarantees convergence of the integrals. Divergence might be an issue when the noise is not sufficiently small in comparison to the smallest distances. With `integral.approximation = 'iteration'`, five iterations is used to determine `m`.

`maxLikLocalDimEst` assumes that the data set is local i.e. a piece of a data set cut out by a sphere with a radius such that the data set is well approximated by a hyperplane (meaning that the curvature should be low in the local data set). See `localIntrinsicDimension`.

## Value

For `maxLikGlobalDimEst` and `maxLikLocalDimEst`, a `DimEst` object with one slot:

`dim.est` the dimension estimate

For `maxLikPointwiseDimEst`, a `DimEstPointwise` object, inheriting `data.frame`, with one slot:

`dim.est` the dimension estimate for each data point. Row `i` has the local dimension estimate at point `data[indices[i], ]`.

## Author(s)

Kerstin Johnsson, Lund University.

## References

Haro, G., Randall, G. and Sapiro, G. (2008) Translated Poisson Mixture Model for Stratification Learning. *Int. J. Comput. Vis.*, **80**, 358-374.

Hill, B. M. (1975) A simple general approach to inference about the tail of a distribution. *Ann. Stat.*, **3**(5) 1163-1174.

Levina, E. and Bickel., P. J. (2005) Maximum likelihood estimation of intrinsic dimension. *Advances in Neural Information Processing Systems 17*, 777-784. MIT Press.

### Examples

```
data <- hyperBall(100, d = 7, n = 13, sd = 0.01)
maxLikGlobalDimEst(data, 10, dnoiseNcChi, 0.01, 13)
maxLikGlobalDimEst(data, 10, dnoiseGaussH, 0.01, 13)
maxLikGlobalDimEst(data, 10, dnoiseGaussH, 0.01, 13)
maxLikGlobalDimEst(data, 10, dnoiseGaussH, 0.01, 13, neighborhood.aggregation = 'robust')
maxLikGlobalDimEst(data, 10, dnoiseGaussH, 0.01, 13,
        integral.approximation = 'guaranteed.convergence',
        neighborhood.aggregation = 'robust')
maxLikGlobalDimEst(data, 10, dnoiseGaussH, 0.01, 13,
        integral.approximation = 'iteration', unbiased = TRUE)

data <- hyperBall(1000, d = 7, n = 13, sd = 0.01)
maxLikGlobalDimEst(data, 500, dnoiseGaussH, 0.01, 13,
        neighborhood.based = FALSE)
maxLikGlobalDimEst(data, 500, dnoiseGaussH, 0.01, 13,
        integral.approximation = 'guaranteed.convergence',
        neighborhood.based = FALSE)
maxLikGlobalDimEst(data, 500, dnoiseGaussH, 0.01, 13,
        integral.approximation = 'iteration',
        neighborhood.based = FALSE)

data <- hyperBall(100, d = 7, n = 13, sd = 0.01)
maxLikPointwiseDimEst(data, 10, dnoiseNcChi, 0.01, 13, indices=1:10)

data <- cutHyperPlane(50, d = 7, n = 13, sd = 0.01)
maxLikLocalDimEst(data, dnoiseNcChi, 0.1, 3)
maxLikLocalDimEst(data, dnoiseGaussH, 0.1, 3)
maxLikLocalDimEst(data, dnoiseNcChi, 0.1, 3,
       integral.approximation = 'guaranteed.convergence')
```

---

## twinPeaks *Twin Peaks*

---

### Description

Generates data points from a two- or higher-dimensional Twin Peaks manifold.

### Usage

```
twinPeaks(Ns, h = 1)
hyperTwinPeaks(Ns, n, h = 1)
```

### Arguments

| | |
|---|---|
| Ns | number of data points. |
| n | dimension of the (hyper) plane from which the peaks stand out. For twinPeaks n is 2. |
| h | height of the peaks. |

## Details

The height of the points is computed as $\prod_1^n \sin(x_i)$, where $x_1, ..., x_n$ are the coordinates of the point in the (hyper) plane.

## Value

A n+1-dimensional data set, where the last dimension represents the height of the points.

## Author(s)

Kerstin Johnsson, Lund University.

## Examples

```
datap <- twinPeaks(400)
par(mfrow = c(1, 3))
plot(datap[,1], datap[,3])
plot(datap[,2], datap[,3])
plot(datap[,1], datap[,2])
```

# Index