Package ‘clusterability’

October 12, 2022

Title Performs Tests for Cluster Tendency of a Data Set

Version 0.1.1.0

Description Test for cluster tendency (clusterability) of a data set.
The methods implemented -
reducing the data set to a single dimension using principal component analysis or computing pairwise distances, and performing a multimodality test like the Dip Test or Silverman's Critical Bandwidth Test -
are described in Adolfsson, Ackerman, and Brownstein (2019) <doi:10.1016/j.patcog.2018.10.026>. Such methods can inform whether clustering algorithms
are appropriate for a data set.

Depends R (>= 3.4.0)

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports diptest, splines

Suggests testthat

NeedsCompilation no

Author Zachariah Neville [aut, cre],
Naomi Brownstein [aut],
Maya Ackerman [aut],
Andreas Adolfsson [aut]

Maintainer Zachariah Neville <z.neville@stat.fsu.edu>

Repository CRAN

Date/Publication 2020-03-04 11:40:07 UTC

R topics documented:

clusterability — *clusterability: a package to perform tests of clusterability*

Description

The clusterabilitytest function can test for clusterability of a dataset, and the print function to display output in the console. Below we include code to use with the provided example datasets. Please see the clusterabilitytest function for documentation on available parameters.

Examples

```
# Normals1
data(normals1)
normals1 <- normals1[,-3]
norm1_dippca <- clusterabilitytest(normals1, "dip")
norm1_dipdist <- clusterabilitytest(normals1, "dip",  distance_standardize = "NONE",
reduction = "distance")
norm1_silvpca <- clusterabilitytest(normals1, "silverman", s_setseed = 123)
norm1_silvdist <- clusterabilitytest(normals1, "silverman", distance_standardize = "NONE",
 reduction = "distance", s_setseed = 123)

print(norm1_dippca)
print(norm1_dipdist)
print(norm1_silvpca)
print(norm1_silvdist)


# Normals2
data(normals2)
normals2 <- normals2[,-3]
norm2_dippca <-
clusterabilitytest(normals2, "dip")
norm2_dipdist <-
clusterabilitytest(normals2, "dip", reduction = "distance", distance_standardize = "NONE")
norm2_silvpca <- clusterabilitytest(normals2, "silverman", s_setseed = 123)
norm2_silvdist <- clusterabilitytest(normals2, "silverman", reduction = "distance",
distance_standardize = "NONE", s_setseed = 123)

print(norm2_dippca)
print(norm2_dipdist)
print(norm2_silvpca)
print(norm2_silvdist)


# Normals3
data(normals3)
normals3 <- normals3[,-3]
norm3_dippca <- clusterabilitytest(normals3, "dip")
norm3_dipdist <- clusterabilitytest(normals3, "dip", reduction = "distance",
distance_standardize = "NONE")
norm3_silvpca <- clusterabilitytest(normals3, "silverman", s_setseed = 123)
norm3_silvdist <- clusterabilitytest(normals3, "silverman", reduction = "distance",
distance_standardize = "NONE", s_setseed = 123)

print(norm3_dippca)
print(norm3_dipdist)
print(norm3_silvpca)
print(norm3_silvdist)



# Normals4
data(normals4)
normals4 <- normals4[,-4]
norm4_dippca <- clusterabilitytest(normals4, "dip")
norm4_dipdist <- clusterabilitytest(normals4, "dip", reduction = "distance",
distance_standardize = "NONE")
norm4_silvpca <- clusterabilitytest(normals4, "silverman", s_setseed = 123)
norm4_silvdist <- clusterabilitytest(normals4, "silverman", reduction = "distance",
distance_standardize = "NONE", s_setseed = 123)

print(norm4_dippca)
print(norm4_dipdist)
print(norm4_silvpca)
print(norm4_silvdist)




# Normals5
data(normals5)
normals5 <- normals5[,-4]
norm5_dippca <- clusterabilitytest(normals5, "dip")
norm5_dipdist <- clusterabilitytest(normals5, "dip", reduction = "distance",
distance_standardize = "NONE")
norm5_silvpca <- clusterabilitytest(normals5, "silverman", s_setseed = 123)
norm5_silvdist <- clusterabilitytest(normals5, "silverman", reduction = "distance",
distance_standardize = "NONE", s_setseed = 123)

print(norm5_dippca)
print(norm5_dipdist)
print(norm5_silvpca)
print(norm5_silvdist)
```

```
# iris
data(iris)
newiris <- iris[,c(1:4)]
iris_dippca <- clusterabilitytest(newiris, "dip")
iris_dipdist <- clusterabilitytest(newiris, "dip", reduction = "distance",
distance_standardize = "NONE")
iris_silvpca <- clusterabilitytest(newiris, "silverman", s_setseed = 123)
iris_silvdist <- clusterabilitytest(newiris, "silverman", reduction = "distance",
 distance_standardize = "NONE", s_setseed = 123)

print(iris_dippca)
print(iris_dipdist)
print(iris_silvpca)
print(iris_silvdist)

# cars
data(cars)

cars_dippca <- clusterabilitytest(cars, "dip")
cars_dipdist <- clusterabilitytest(cars, "dip", reduction = "distance",
distance_standardize = "NONE")
cars_silvpca <- clusterabilitytest(cars, "silverman", s_setseed = 123)
cars_silvdist <- clusterabilitytest(cars, "silverman", reduction = "distance",
distance_standardize = "NONE", s_setseed = 123)

print(cars_dippca)
print(cars_dipdist)
print(cars_silvpca)
print(cars_silvdist)
```

`clusterabilitytest` *Perform a test of clusterability*

Description

Performs tests for clusterability of a data set and returns results in a clusterability object. Can do data reduction via PCA or pairwise distances and standardize data prior to performing the test.

Usage

```
clusterabilitytest(data, test, reduction = "pca",
  distance_metric = "euclidean", distance_standardize = "std",
  pca_center = TRUE, pca_scale = TRUE, is_dist_matrix = FALSE,
  completecase = FALSE, d_simulatepvalue = FALSE, d_reps = 2000,
  s_m = 999, s_adjust = TRUE, s_digits = 6, s_setseed = NULL,
  s_outseed = FALSE)
```

Arguments

`data` the data set to be used in the test. Must contain only numeric data.

`test` the test to be performed. Either `"dip"` or `"silverman"`. See 'Details' section below for how to pick a test.

`reduction` any dimension reduction that is to be performed.

- `"none"` performs no dimension reduction.
- `"pca"` uses the scores from the first principal component.
- `"distance"` computes pairwise distances (using `distance_metric` as the metric).

For multivariate `data`, dimension reduction is required.

`distance_metric`

if applicable, the metric to be used in computing pairwise distances.

The `"euclidean"` (default), `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"` choices work the same as in `dist`. The Minkowski metric is available by providing `"minkowski(p)"`.

Additional choices are:

- `"sqeuc"`: squared Euclidean distances.
- `"cov"`: covariance similarity coefficient,
- `"corr"`: correlation similarity coefficient
- `"sqcorr"`: squared correlation similarity coefficient.

CAUTION: Not all of these have been tested, but instead are provided to potentially be useful. If in doubt, use the default `"euclidean"`.

`distance_standardize`

how the variables should be standardized, if at all.

- `"none"`: no standardization is performed
- `"std"` (default) each variable standardized to have mean 0 and standard deviation 1
- `"mean"`: each variable standardized to have mean 0 (standard deviation is unchanged)
- `"median"`: each variable standardized to have median 0 (standard deviation is unchanged)

`pca_center` if applicable, a logical value indicating whether the variables should be shifted to be zero centered (see `prcomp` for more details). Default is `TRUE`.

`pca_scale` if applicable, a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place (see `prcomp` for details). Default is `TRUE`.

`is_dist_matrix` a logical value indicating whether the `data` argument is a distance matrix. If `TRUE` then the lower triangular portion of `data` will be extracted and be used in the multimodality test.

`completecase` a logical value indicating whether a complete case analysis should be performed. For both tests, missing data must be removed before the test can be performed. This can be done manually by the user or by setting `completecase = TRUE`.

`d_simulatepvalue`
: for Dip Test, a logical value indicating whether p~values should be obtained via Monte Carlo simulation (see `dip.test` for details).

`d_reps`
: for Dip Test, a positive integer. The number of replicates used in Monte Carlo simulation. Only used if `d_simulatepvalue` is `TRUE`.

`s_m`
: for Silverman Test, a positive integer. The number of bootstrap replicates used in the test. Default is `999`.

`s_adjust`
: for Silverman Test, a logical value indicating whether p-values are adjusted using work by Hall and York.

`s_digits`
: for Silverman Test, a positive integer indicating the number of digits to round the p value. Default is `6` and is only used when `s_adjust = TRUE`.

`s_setseed`
: for Silverman Test, an integer used to set the seed of the random number generator. If the default value of `NULL` is used, then no seed will be set.

`s_outseed`
: for Silverman Test, a logical value indicating whether to return the state of the random number generator as part of the output. This is used in limited cases for troubleshooting, so the default is `FALSE`.

Value

`clusterabilitytest` returns a `clusterability` object containing information on the test performed and results. Can be printed using the `print.clusterability` function.

References

Hall, P. and York, M., 2001. On the calibration of Silverman's test for multimodality. Statistica Sinica, pp.515-536.

Silverman, B.W., 1981. Using kernel density estimates to investigate multimodality. Journal of the Royal Statistical Society. Series B (Methodological), pp.97-99.

Martin Maechler (2016). diptest: Hartigan's Dip Test Statistic for Unimodality - Corrected. R package version 0.75-7. https://CRAN.R-project.org/package=diptest

Schwaiger F, Holzmann H. Package which implements the silvermantest; 2013. Available from: https://www.mathematik.uni-marburg.de/stochastik/R packages/.

See Also

`print.clusterability`

Examples

```
### Quick start ###
# Load data and remove Species
data(iris)
iris_num <- iris[,-5]
plot(iris_num)

# Run test using default options
clust_result <- clusterabilitytest(iris_num, "dip")
```

```
# Print results
print(clust_result)


### Longer Example: Specifying Parameters ###
# Load data and plot to visualize
data(normals2)
plot(normals2)

# Using Silverman's test, pairwise distances to reduce dimension,
# 1,000 bootstrap replicates, with an RNG seed of 12345
clust_result2 <- clusterabilitytest(normals2, "silverman", reduction = "distance",
     s_m = 1000, s_setseed = 12345)

# Print result
print(clust_result2)
```

`normals1` *Data generated from a single multivariate Normal distribution, 2 dimensions.*

Description

A dataset containing 150 observations generated from a multivariate Normal distribution. The distribution has mean vector (0, 4), each variable has unit variance, and the variables are uncorrelated. This dataset is not clusterable.

Usage

```
normals1
```

Format

A data frame with 150 rows and 3 variables:

x x variable

y y variable

cluster Distribution from which the observation was sampled

Details

The cluster variable is 1 for all observations because all were sampled from the same distribution. Remove the variable before using the dataset in any tests.

`normals2` *Data generated from a mixture of two multivariate Normal distributions, 2 dimensions. A dataset containing 150 observations generated from a mixture of two multivariate Normal distributions. 75 observations come from a distribution with mean vector (-3, -2) with each variable having unit variance and uncorrelated with each other. 75 observations come from a distribution with mean vector (1, 1) with each variable having unit variance and uncorrelated with each other. The dataset is clusterable.*

Description

Remove the cluster variable before using the dataset in any tests.

Usage

```
normals2
```

Format

A data frame with 150 rows and 3 variables:

x x variable

y y variable

cluster Distribution from which the observation was sampled

`normals3` *Data generated from a mixture of three multivariate Normal distributions, 2 dimensions. A dataset containing 150 observations generated from a mixture of three multivariate Normal distributions. 50 observations are from a distribution with mean vector (3, 0), 50 observations from a distribution with mean vector (0, 3), and 50 observations from a distribution with mean vector (3, 6). For each of these three distributions, the x and y variables have unit variance and are uncorrelated. The dataset is clusterable.*

Description

Remove the cluster variable before using the dataset in any tests.

Usage

```
normals3
```

Format

A data frame with 150 rows and 3 variables:

- **x** x variable
- **y** y variable
- **cluster** Distribution from which the observation was sampled

---

`normals4` *Data generated from a mixture of two multivariate Normal distributions, 3 dimensions. A dataset containing 150 observations generated from a mixture of two multivariate Normal distributions. 75 observations come from a distribution with mean vector (1, 3, 2) and 75 observations come from a distribution with mean vector (4, 6, 0). For each distribution, the variables each have unit variance and are uncorrelated. The dataset is clusterable.*

Description

Remove the cluster variable before using the dataset in any tests.

Usage

```
normals4
```

Format

A data frame with 150 rows and 4 variables:

- **x** x variable
- **y** y variable
- **z** z variable
- **cluster** Distribution from which the observation was sampled

normals5 — *Data generated from a mixture of three multivariate Normal distributions, 3 dimensions. A dataset containing 150 observations generated from a mixture of three multivariate Normal distributions. 50 observations come from a distribution with mean vector (1, 3, 3), 50 observations come from a distribution with mean vector (4, 6, 0), and 50 observations come from a distribution with mean vector (2, 8, -3). For each distribution, the variables each have unit variance and are uncorrelated. The dataset is clusterable.*

Description

Remove the cluster variable before using the dataset in any tests.

Usage

```
normals5
```

Format

A data frame with 150 rows and 4 variables:

x x variable

y y variable

z z variable

cluster Distribution from which the observation was sampled

print.clusterability — *Print a clusterability object*

Description

Print function to display results from a clusterability test.

Usage

```
## S3 method for class 'clusterability'
print(x, ...)
```

Arguments

`x`	An object of class ’clusterability’
`...`	Not used

See Also

clusterabilitytest

Index