Package 'clusteval'

August 29, 2016

Title Evaluation of Clustering Algorithms

Version 0.1

Date 2012-08-30

Author John A. Ramey

Maintainer John A. Ramey <johnramey@gmail.com>

Description An R package that provides a suite of tools to evaluate clustering algorithms, clusterings, and individual clusters.

Depends R (>= 2.15)

Imports parallel, mvtnorm, Rcpp (>= 0.9.13)

LinkingTo Rcpp

License MIT

Collate 'similarity.r' 'sim_normal.r' 'sim_unif.r' 'sim_student.r' 'clustomit.r' 'sim_data.r' 'helper-boot.r' 'clusteval.r' 'random_clustering.r' 'comembership.r'

Repository CRAN

Date/Publication 2012-08-31 17:17:52

NeedsCompilation yes

R topics documented:

boot_stratified_omit 2
cluster_similarity 3
clusteval 4
clustomit 4
comembership 6
comembership_table 7
intraclass_cov 9
jaccard_indep 9
random_clustering 11
rand_indep 12
sim_data 13

sim_normal 14
sim_student 15
sim_unif 17

Index **19**

`boot_stratified_omit` *Creates a list of indices for a stratified nonparametric bootstrap.*

Description

This function creates a list of indices for a stratified nonparametric bootstrap. Corresponding to our Cluster Omission Stability statistic implemented in `clustomit`, we omit each group in turn and perform a stratified bootstrap without the group. We denote the number of groups as `num_clusters`, which is equal to `nlevels(factor(y))`. Specifically, suppose that we omit the kth group. That is, we ignore all of the observations corresponding to group k. Then, we sample with replacement from each of the remaining groups (i.e., every group except for group k), yielding a set of bootstrap indices.

Usage

```
boot_stratified_omit(y, num_reps = 50)
```

Arguments

`y`	a vector that denotes the grouping of each observation. It must be coercible with `as.factor`.
`num_reps`	the number of bootstrap replications to use for each group

Details

The returned list contains $K \times num_reps$ elements.

Value

named list containing indices for each bootstrap replication

Examples

```
set.seed(42)
# We use 4 clusters, each with up to 10 observations. The sample sizes are
# randomly chosen.
num_clusters <- 4
sample_sizes <- sample(10, num_clusters, replace = TRUE)

# Create the cluster labels, y.
y <- unlist(sapply(seq_len(num_clusters), function(k) {
 rep(k, sample_sizes[k])
}))
```

```
# Use 20 reps per group.
boot_stratified_omit(y, num_reps = 20)

# Use the default number of reps per group.
boot_stratified_omit(y)
```

`cluster_similarity` *Computes the similarity between two clusterings of the same data set.*

Description

For two clusterings of the same data set, this function calculates the similarity statistic specified of the clusterings from the comemberships of the observations. Basically, the comembership is defined as the pairs of observations that are clustered together.

Usage

```
cluster_similarity(labels1, labels2,
  similarity = c("jaccard", "rand"),
  method = "independence")
```

Arguments

`labels1`	a vector of n clustering labels
`labels2`	a vector of n clustering labels
`similarity`	the similarity statistic to calculate
`method`	the model under which the statistic was derived

Details

To calculate the similarity, we compute the 2x2 contingency table, consisting of the following four cells:

n_11 the number of observation pairs where both observations are comembers in both clusterings

n_10 the number of observation pairs where the observations are comembers in the first clustering but not the second

n_01 the number of observation pairs where the observations are comembers in the second clustering but not the first

n_00 the number of observation pairs where neither pair are comembers in either clustering

Currently, we have implemented the following similarity statistics:

- Rand index
- Jaccard coefficient

To compute the contingency table, we use the `comembership_table` function.

Value

the similarity between the two clusterings

Examples

```
# Notice that the number of comemberships is 'n choose 2'.
iris_kmeans <- kmeans(iris[, -5], centers = 3)$cluster
iris_hclust <- cutree(hclust(dist(iris[, -5])), k = 3)
cluster_similarity(iris_kmeans, iris_hclust)
```

---

clusteval *Evaluation of Clustering Algorithms*

Description

An R package that provides a suite of tools to evaluate clustering algorithms, clusterings, and individual clusters.

---

clustomit *ClustOmit - Cluster Stability Evaluation via Cluster Omission*

Description

We provide an implementation of the ClustOmit statistic, which is an approach to evaluating the stability of a clustering determined by a clustering algorithm. As discussed by Hennig (2007), arguably a stable clustering is one in which a perturbation of the original data should yield a similar clustering. However, if a perturbation of the data yields a large change in the clustering, the original clustering is considered unstable. The ClustOmit statistic provides an approach to detecting instability via a stratified, nonparametric resampling scheme. We determine the stability of the clustering via the similarity statistic specified (by default, the Jaccard coefficient).

Usage

```
clustomit(x, num_clusters, cluster_method,
  similarity = c("jaccard", "rand"),
  weighted_mean = TRUE, num_reps = 50,
  num_cores = getOption("mc.cores", 2), ...)
```

Arguments

x	data matrix with n observations (rows) and p features (columns)
num_clusters	the number of clusters to find with the clustering algorithm specified in cluster_method
cluster_method	a character string or a function specifying the clustering algorithm that will be used. The method specified is matched with the match.fun function. The function given should return only clustering labels for each observation in the matrix x.
similarity	the similarity statistic that is used to compare the original clustering (after a single cluster and its observations have been omitted) to its resampled counterpart. Currently, we have implemented the Jaccard and Rand similarity statistics and use the Jaccard statistic by default.
weighted_mean	logical value. Should the aggregate similarity score for each bootstrap replication be weighted by the number of observations in each of the observed clusters? By default, yes (i.e., TRUE).
num_reps	the number of bootstrap replicates to draw for each omitted cluster
num_cores	the number of coures to use. If 1 core is specified, then lapply is used without parallelization. See the mc.cores argument in mclapply for more details.
...	additional arguments passed to the function specified in cluster_method

Details

To compute the ClustOmit statistic, we first cluster the data given in x into num_clusters clusters with the clustering algorithm specified in cluster_method. We then omit each cluster in turn and all of the observations in that cluster. For the omitted cluster, we resample from the remaining observations and cluster the resampled observations into num_clusters - 1 clusters again using the clustering algorithm specified in cluster_method. Next, we compute the similarity between the cluster labels of the original data set and the cluster labels of the bootstrapped sample. We approximate the sampling distribution of the ClustOmit statistic using a stratified, nonparametric bootstrapping scheme and use the apparent variability in the approximated sampling distribution as a diagnostic tool for further evaluation of the proposed clusters. By default, we utilize the Jaccard similarity coefficient in the calculation of the ClustOmit statistic to provide a clear interpretation of cluster assessment. The technical details of the ClustOmit statistic can be found in our forthcoming publication entitled "Cluster Stability Evaluation of Gene Expression Data."

The ClustOmit cluster stability statistic is based on the cluster omission admissibility condition from Fisher and Van Ness (1971), who provide decision-theoretic admissibility conditions that a reasonable clustering algorithm should satisfy. The guidelines from Fisher and Van Ness (1971) establish a systematic foundation that is often lacking in the evaluation of clustering algorithms. The ClustOmit statistic is our proposed methodology to evaluate the cluster omission admissibility condition from Fisher and Van Ness (1971).

We require a clustering algorithm function to be specified in the argument cluster_method. The function given should accept at least two arguments:

x matrix of observations to cluster

num_clusters the number of clusters to find

... additional arguments that can be passed on

Also, the function given should return only clustering labels for each observation in the matrix `x`. The additional arguments specified in `...` are useful if a wrapper function is used: see the example below for an illustration.

Value

object of class `clustomit`, which contains a named list with elements

boot_aggregate: vector of the aggregated similarity statistics for each bootstrap replicate

boot_similarity: list containing the bootstrapped similarity scores for each cluster omitted

obs_clusters: the clustering labels determined for the observations in `x`

num_clusters: the number of clusters found

similarity: the similarity statistic used for comparison between the original clustering and the re-sampled clusterings

References

Fisher, L. and Van Ness, J. (1971), Admissible Clustering Procedures, _Biometrika_, 58, 1, 91-104.

Hennic, C. (2007), Cluster-wise assessment of cluster stability, _Computational Statistics and Data Analysis_, 52, 258-271. http://www.jstor.org/stable/2334320

Examples

```
# First, we create a wrapper function for the K-means clustering algorithm
# that returns only the clustering labels for each observation (row) in
# \code{x}.
kmeans_wrapper <- function(x, num_clusters, num_starts = 10, ...) {
  kmeans(x = x, centers = num_clusters, nstart = num_starts, ...)$cluster
}

# For this example, we generate five multivariate normal populations with the
# \code{sim_data} function.
x <- sim_data("normal", delta = 1.5, seed = 42)$x

clustomit_out <- clustomit(x = x, num_clusters = 4,
                           cluster_method = "kmeans_wrapper", num_cores = 1)
clustomit_out2 <- clustomit(x = x, num_clusters = 5,
                            cluster_method = kmeans_wrapper, num_cores = 1)
```

`comembership` *Calculates the comemberships of all pairs of a vector of clustering labels.*

Description

For a set of clustering labels, this function computes the comembership of all pairs of observations. Basically, two observations are said to be comembers if they are clustered together.

Usage

```
comembership(labels)
```

Arguments

labels	a vector of n clustering labels

Details

Tibshirani and Walther (2005) use the term 'co-membership', which we shorten to 'comembership'. Some authors instead use the terms 'connectivity' or 'co-occurrence'.

We use the Rcpp package to improve the runtime speed of this function.

Value

a vector of choose(n, 2) comembership bits

References

Tibshirani, R. and Walther, G. (2005), Cluster Validation by Prediction Strength, _Journal of Computational and Graphical Statistics_, 14, 3, 511-528. http://amstat.tandfonline.com/doi/abs/10.1198/106186005X59243.

Examples

```
# We generate K = 3 labels for each of n = 10 observations and compute the
# comembership for all 'n choose 2' pairs.
set.seed(42)
K <- 3
n <- 10
labels <- sample.int(K, n, replace = TRUE)
comembership_out <- comembership(labels)
comembership_out

# Notice that the number of comemberships is 'n choose 2'.
length(comembership_out) == choose(n, 2)
```

---

comembership_table *Calculates the 2x2 contingency table of agreements and disagreements of comemberships from two vectors of clustering labels.*

Description

For two clusterings of the same data set, this function calculates the 2x2 contingency table of agreements and disagreements of the corresponding two vectors of comemberships. Basically, the comembership is defined as the pairs of observations that are clustered together.

Usage

```
comembership_table(labels1, labels2)
```

Arguments

labels1	a vector of n clustering labels
labels2	a vector of n clustering labels

Details

The contingency table calculated is typically utilized in the calculation of a similarity statistic (e.g., Rand index, Jaccard index) between the two clusterings. The 2x2 contingency table consists of the following four cells:

n_11 the number of observation pairs where both observations are comembers in both clusterings

n_10 the number of observation pairs where the observations are comembers in the first clustering but not the second

n_01 the number of observation pairs where the observations are comembers in the second clustering but not the first

n_00 the number of observation pairs where neither pair are comembers in either clustering

Tibshirani and Walther (2005) use the term 'co-membership', which we shorten to 'comembership'. Some authors instead use the terms 'connectivity' or 'co-occurrence'.

We use the Rcpp package to improve the runtime speed of this function.

Value

named list containing the calculated contingency table:

- n_11
- n_10
- n_01
- n_00

References

Tibshirani, R. and Walther, G. (2005). Cluster Validation by Prediction Strength. Journal of Computational and Graphical Statistics, 14, 3, 511-528. http://amstat.tandfonline.com/doi/abs/10.1198/106186005X59243.

Examples

```
# We generate K = 3 labels for each of n = 10 observations and compute the
# comembership for all 'n choose 2' pairs.
set.seed(42)
K <- 3
n <- 10
labels1 <- sample.int(K, n, replace = TRUE)
```

```
labels2 <- sample.int(K, n, replace = TRUE)
comembership_table(labels1, labels2)

# Here, we cluster the \code{\link{iris}} data set with the K-means and
# hierarchical algorithms using the true number of clusters, K = 3.
# Then, we compute the 2x2 contingency table agreements and disagreements of
#' the comemberships.
iris_kmeans <- kmeans(iris[, -5], centers = 3)$cluster
iris_hclust <- cutree(hclust(dist(iris[, -5])), k = 3)
comembership_table(iris_kmeans, iris_hclust)
```

`intraclass_cov` *Construct an intraclass covariance matrix.*

Description

We define a $p \times p$ intraclass covariance (correlation) matrix to be $\Sigma_m = \sigma^2(1-\rho)J_p + \rho I_p$, where $-(p-1)^{-1} < \rho < 1$, I_p is the $p \times p$ identity matrix, and J_p denotes the $p \times p$ matrix of ones.

Usage

```
intraclass_cov(p, rho, sigma2 = 1)
```

Arguments

`p`	the dimension of the matrix
`rho`	the intraclass covariance (correlation) constant
`sigma2`	the coefficient of the intraclass covariance matrix

Value

an intraclass covariance matrix matrix of size p

`jaccard_indep` *Computes the Jaccard similarity coefficient of two clusterings of the same data set under the assumption that the two clusterings are independent.*

Description

For two clusterings of the same data set, this function calculates the Jaccard similarity coefficient of the clusterings from the comemberships of the observations. Basically, the comembership is defined as the pairs of observations that are clustered together.

Usage

```
jaccard_indep(labels1, labels2)
```

Arguments

labels1	a vector of n clustering labels
labels2	a vector of n clustering labels

Details

To calculate the Rand index, we compute the 2x2 contingency table, consisting of the following four cells:

n_11 the number of observation pairs where both observations are comembers in both clusterings

n_10 the number of observation pairs where the observations are comembers in the first clustering but not the second

n_01 the number of observation pairs where the observations are comembers in the second clustering but not the first

n_00 the number of observation pairs where neither pair are comembers in either clustering

The Jaccard similarity coefficient is defined as:

$$J = \frac{n_{11}}{n_{11} + n_{10} + n_{01}}$$

.

In the special case that the Jaccard coefficient results in $0/0$, we define $J = 0$. For instance, this case can occur when both clusterings consist of all singleton clusters.

To compute the contingency table, we use the comembership_table function.

Value

the Jaccard coefficient for the two sets of cluster labels (See Details.)

Examples

```
## Not run:
# We generate K = 3 labels for each of n = 10 observations and compute the
# Jaccard similarity coefficient between the two clusterings.
set.seed(42)
K <- 3
n <- 10
labels1 <- sample.int(K, n, replace = TRUE)
labels2 <- sample.int(K, n, replace = TRUE)
jaccard_indep(labels1, labels2)

# Here, we cluster the \code{\link{iris}} data set with the K-means and
# hierarchical algorithms using the true number of clusters, K = 3.
# Then, we compute the Jaccard similarity coefficient between the two
# clusterings.
iris_kmeans <- kmeans(iris[, -5], centers = 3)$cluster
iris_hclust <- cutree(hclust(dist(iris[, -5])), k = 3)
jaccard_indep(iris_kmeans, iris_hclust)

## End(Not run)
```

`random_clustering` *Randomly cluster a data set into K clusters.*

Description

For each observation (row) in 'x', one of K labels is randomly generated. By default, the probabilities of selecting each clustering label are equal, but this can be altered by specifying 'prob', a vector of probabilities for each cluster.

Usage

```
random_clustering(x, K, prob = NULL)
```

Arguments

`x`	a matrix containing the data to cluster. The rows are the sample observations, and the columns are the features.
`K`	the number of clusters
`prob`	a vector of probabilities to generate each cluster label. If NULL, each cluster label has an equal chance of being selected.

Details

Random clustering is often utilized as a baseline comparison clustering against which other clustering algorithms are employed to identify structure within the data. Typically, comparisons are made in terms of proposed clustering assessment and evaluation methods as well as clustering similarity measures. For the former, a specified clustering evaluation method is computed for the considered clustering algorithms as well as random clustering. If the clusters determined by a considered clustering algorithm do not differ significantly from the random clustering, we might conclude that the found clusters are no better than naively choosing clustering labels for each observation at random. Likewise, a similarity measure can be computed to compare the clusterings from each of a considered clustering algorithm and a random clustering: if the clusterings are significantly similar, once again, we might conclude the clusters found via the considered clustering algorithm do not differ significantly from those found at random. In either case, the clusters are unlikely to provide meaningful results on which the user can better understand the inherent structure within the data.

Value

a vector of clustering labels for each observation in 'x'.

`rand_indep` *Computes the Rand similarity index of two clusterings of the same data set under the assumption that the two clusterings are independent.*

Description

For two clusterings of the same data set, this function calculates the Rand similarity coefficient of the clusterings from the comemberships of the observations. Basically, the comembership is defined as the pairs of observations that are clustered together.

Usage

```
rand_indep(labels1, labels2)
```

Arguments

`labels1`	a vector of `n` clustering labels
`labels2`	a vector of `n` clustering labels

Details

To calculate the Rand index, we compute the 2x2 contingency table, consisting of the following four cells:

n_11 the number of observation pairs where both observations are comembers in both clusterings

n_10 the number of observation pairs where the observations are comembers in the first clustering but not the second

n_01 the number of observation pairs where the observations are comembers in the second clustering but not the first

n_00 the number of observation pairs where neither pair are comembers in either clustering

The Rand similarity index is defined as:

$$R = \frac{n_{11} + n_{00}}{n_{11} + n_{10} + n_{01} + n_{00}}$$

.

To compute the contingency table, we use the `comembership_table` function.

Value

the Rand index for the two sets of cluster labels

Examples

```
## Not run:
# We generate K = 3 labels for each of n = 10 observations and compute the
# Rand similarity index between the two clusterings.
set.seed(42)
K <- 3
n <- 10
labels1 <- sample.int(K, n, replace = TRUE)
labels2 <- sample.int(K, n, replace = TRUE)
rand_indep(labels1, labels2)

# Here, we cluster the \code{\link{iris}} data set with the K-means and
# hierarchical algorithms using the true number of clusters, K = 3.
# Then, we compute the Rand similarity index between the two clusterings.
iris_kmeans <- kmeans(iris[, -5], centers = 3)$cluster
iris_hclust <- cutree(hclust(dist(iris[, -5])), k = 3)
rand_indep(iris_kmeans, iris_hclust)

## End(Not run)
```

---

sim_data *Wrapper function to generate data from a variety of data-generating models.*

Description

We provide a wrapper function to generate from three data-generating models:

`sim_unif` Five multivariate uniform distributions

`sim_normal` Multivariate normal distributions with intraclass covariance matrices

`sim_student` Multivariate Student's t distributions each with a common covariance matrix

Usage

```
sim_data(family = c("uniform", "normal", "student"), ...)
```

Arguments

`family`	the family of distributions from which to generate data
`...`	optional arguments that are passed to the data-generating function

Details

For each data-generating model, we generate n_m observations $(m = 1, \ldots, M)$ from each of M multivariate distributions so that the Euclidean distance between each of the population centroids and the origin is equal and scaled by $\Delta \geq 0$. For each model, the argument `delta` controls this separation.

This wrapper function is useful for simulation studies, where the efficacy of supervised and unsupervised learning methods and algorithms are evaluated as a the population separation is increased.

Value

named list containing:

x: A matrix whose rows are the observations generated and whose columns are the p features (variables)

y: A vector denoting the population from which the observation in each row was generated.

Examples

```
set.seed(42)
uniform_data <- sim_data(family = "uniform")
normal_data <- sim_data(family = "normal", delta = 2)
student_data <- sim_data(family = "student", delta = 1, df = 1:5)
```

`sim_normal` *Generates random variates from multivariate normal populations with intraclass covariance matrices.*

Description

We generate n_m observations $(m = 1, \ldots, M)$ from each of M multivariate normal distributions such that the Euclidean distance between each of the means and the origin is equal and scaled by $\Delta \geq 0$.

Usage

```
sim_normal(n = rep(25, 5), p = 50, rho = rep(0.9, 5),
  delta = 0, sigma2 = 1, seed = NULL)
```

Arguments

`n`	a vector (of length M) of the sample sizes for each population
`p`	the dimension of the multivariate normal populations
`rho`	a vector (of length M) of the intraclass constants for each population
`delta`	the fixed distance between each population and the origin
`sigma2`	the coefficient of each intraclass covariance matrix
`seed`	seed for random number generation (If NULL, does not set seed)

Details

Let Π_m denote the mth population with a p-dimensional multivariate normal distribution, $N_p(\mu_m, \Sigma_m)$ with mean vector μ_m and covariance matrix Σ_m. Also, let e_m be the mth standard basis vector (i.e., the mth element is 1 and the remaining values are 0). Then, we define

$$\mu_m = \Delta \sum_{j=1}^{p/M} e_{(p/M)(m-1)+j}.$$

Note that p must be divisible by M. By default, the first 10 dimensions of μ_1 are set to delta with all remaining dimensions set to 0, the second 10 dimensions of μ_2 are set to delta with all remaining dimensions set to 0, and so on.

Also, we consider intraclass covariance (correlation) matrices such that $\Sigma_m = \sigma^2(1-\rho_m)J_p + \rho_m I_p$, where $-(p-1)^{-1} < \rho_m < 1$, I_p is the $p \times p$ identity matrix, and J_p denotes the $p \times p$ matrix of ones.

By default, we let $M = 5$, $\Delta = 0$, and $\sigma^2 = 1$. Furthermore, we generate 25 observations from each population by default.

For $\Delta = 0$ and $\rho_m = \rho$, $m = 1, \ldots, M$, the M populations are equal.

Value

named list containing:

x: A matrix whose rows are the observations generated and whose columns are the p features (variables)

y: A vector denoting the population from which the observation in each row was generated.

Examples

```
data_generated <- sim_normal(n = 10 * seq_len(5), seed = 42)
dim(data_generated$x)
table(data_generated$y)

data_generated2 <- sim_normal(p = 10, delta = 2, rho = rep(0.1, 5))
table(data_generated2$y)
sample_means <- with(data_generated2,
                     tapply(seq_along(y), y, function(i) {
                            colMeans(x[i,])
                     }))
(sample_means <- do.call(rbind, sample_means))
```

---

sim_student *Generates random variates from multivariate Student's t populations.*

Description

We generate n_m observations $(m = 1, \ldots, M)$ from each of M multivariate Student's t distributions such that the Euclidean distance between each of the means and the origin is equal and scaled by $\Delta \geq 0$.

Usage

```
sim_student(n = rep(25, 5), p = 50, df = rep(6, 5),
  delta = 0, Sigma = diag(p), seed = NULL)
```

Arguments

n	a vector (of length M) of the sample sizes for each population
p	the dimension of the multivariate Student's t distributions
df	a vector (of length M) of the degrees of freedom for each population
delta	the fixed distance between each population and the origin
Sigma	the common covariance matrix
seed	seed for random number generation (If NULL, does not set seed)

Details

Let Π_m denote the mth population with a p-dimensional multivariate Student's t distribution, $T_p(\mu_m, \Sigma_m, c_m)$, where μ_m is the population location vector, Σ_m is the positive-definite covariance matrix, and c_m is the degrees of freedom.

Let e_m be the mth standard basis vector (i.e., the mth element is 1 and the remaining values are 0). Then, we define

$$\mu_m = \Delta \sum_{j=1}^{p/M} e_{(p/M)(m-1)+j}.$$

Note that p must be divisible by M. By default, the first 10 dimensions of μ_1 are set to delta with all remaining dimensions set to 0, the second 10 dimensions of μ_2 are set to delta with all remaining dimensions set to 0, and so on.

We use a common covariance matrix $\Sigma_m = \Sigma$ for all populations.

For small values of c_m, the tails are heavier, and, therefore, the average number of outlying observations is increased.

By default, we let $M = 5$, $\Delta = 0$, $\Sigma_m = I_p$, and $c_m = 6$, $m = 1, \ldots, M$, where I_p denotes the $p \times p$ identity matrix. Furthermore, we generate 25 observations from each population by default.

For $\Delta = 0$ and $c_m = c$, $m = 1, \ldots, M$, the M populations are equal.

Value

named list containing:

x: A matrix whose rows are the observations generated and whose columns are the p features (variables)

y: A vector denoting the population from which the observation in each row was generated.

Examples

```
data_generated <- sim_student(n = 10 * seq_len(5), seed = 42)
dim(data_generated$x)
table(data_generated$y)

data_generated2 <- sim_student(p = 10, delta = 2, df = rep(2, 5))
table(data_generated2$y)
sample_means <- with(data_generated2,
                     tapply(seq_along(y), y, function(i) {
```

```
                        colMeans(x[i,])
                    }))
(sample_means <- do.call(rbind, sample_means))
```

`sim_unif` *Generates random variates from five multivariate uniform populations.*

Description

We generate `n` observations from each of four trivariate distributions such that the Euclidean distance between each of the populations is a fixed constant, `delta` > 0.

Usage

```
sim_unif(n = rep(25, 5), delta = 0, seed = NULL)
```

Arguments

`n`	a vector (of length M = 5) of the sample sizes for each population
`delta`	the fixed distance between each population and the origin
`seed`	Seed for random number generation. (If NULL, does not set seed)

Details

To define the populations, let $x = (X_1, \ldots, X_p)'$ be a multivariate uniformly distributed random vector such that $X_j \sim U(a_j, b_j)$ is an independently distributed uniform random variable with $a_j < b_j$ for $j = 1, \ldots, p$. Let Pi_m denote the mth population ($m = 1, \ldots, 5$). Then, we have the five populations:

$$\Pi_1 = U(-1/2, 1/2) \times U(\Delta - 1/2, \Delta + 1/2) \times U(-1/2, 1/2) \times U(-1/2, 1/2),$$

$$\Pi_2 = U(\Delta - 1/2, \Delta + 1/2) \times U(-1/2, 1/2) \times U(-1/2, 1/2) \times U(-1/2, 1/2),$$

$$\Pi_3 = U(-1/2, 1/2) \times U(-\Delta - 1/2, -\Delta + 1/2) \times U(-1/2, 1/2) \times U(-1/2, 1/2),$$

$$\Pi_4 = U(-1/2, 1/2) \times U(-1/2, 1/2) \times U(-\Delta - 1/2, -\Delta + 1/2) \times U(-1/2, 1/2),$$

$$\Pi_5 = U(-1/2, 1/2) \times U(-1/2, 1/2) \times U(-1/2, 1/2) \times U(\Delta - 1/2, \Delta + 1/2).$$

We generate n_m observations from population Π_m.

For $\Delta = 0$ and $\rho_m = \rho$, $m = 1, \ldots, M$, the M populations are equal.

Notice that the support of each population is a unit hypercube with 4 features. Moreover, for $\Delta \geq 1$, the populations are mutually exclusive and entirely separated.

Value

named list containing:

x: A matrix whose rows are the observations generated and whose columns are the p features (variables)

y: A vector denoting the population from which the observation in each row was generated.

Examples

```
data_generated <- sim_unif(seed = 42)
dim(data_generated$x)
table(data_generated$y)

data_generated2 <- sim_unif(n = 10 * seq_len(5), delta = 1.5)
table(data_generated2$y)
sample_means <- with(data_generated2,
                     tapply(seq_along(y), y, function(i) {
                            colMeans(x[i,])
                     }))
(sample_means <- do.call(rbind, sample_means))
```

Index

boot_stratified_omit, 2

cluster_similarity, 3
clusteval, 4
clusteval-package *(clusteval)*, 4
clustomit, *2*, 4
comembership, 6
comembership_table, *3*, 7, *10*, *12*

intraclass_cov, 9

jaccard_indep, 9

lapply, *5*

match.fun, *5*
mclapply, *5*

package-clusteval *(clusteval)*, 4

rand_indep, 12
random_clustering, 11

sim_data, 13
sim_normal, *13*, 14
sim_student, *13*, 15
sim_unif, *13*, 17