Package: SampleCore (via r-universe)

May 29, 2026

Title Sampling Strategies for Constructing Core Collections

Version 0.1.0.9000

Description Implements multiple allocation and selection strategies of sampling to construct core collections primarily from clustered or grouped germplasm collection data. Provides methods for allocating entries to clusters/groups based on group sizes, group-wise distance-based diversity metrics, and group-wise diversity index estimates. Includes procedures for selecting entries within clusters/groups through random sampling, genetic distance-based approaches, and optimized diversity metric–based selection methods. See the package documentation for more, including full list of references for the methods implemented.

License GPL (>= 2)

Encoding UTF-8

BuildManual TRUE

Imports cluster, DiversityStats, ggplot2, igraph, MASS, mathjaxr, prospectr, Rdpack, Rtsne, stats, vegan

Suggests biotools, dbscan, EvaluateCore, fastcluster, knitr, pander, rmarkdown

RdMacros mathjaxr, Rdpack

Copyright 2024-2026, ICAR-NBPGR

URL https://cran.r-project.org/package=SampleCore, https://github.com/aravind-j/SampleCore, https://doi.org/10.5281/zenodo.20449499, https://aravind-j.github.io/SampleCore/

BugReports https://github.com/aravind-j/SampleCore/issues

Depends R (>= 3.5)

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Config/pak/sysreqs libglpk-dev libicu-dev libxml2-dev

Repository https://aravind-j.r-universe.dev

Date/Publication 2026-05-29 18:03:03 UTC

RemoteUrl https://github.com/aravind-j/samplecore

RemoteRef HEAD

RemoteSha f9ca14b6fefdca50223de8ff93933c37d9e4c284

Contents

`allocate.basic` *Allocation of Entries to be Selected from Clusters/Groups based on Size for Core Collection Development*

Description

Estimate the number of entries to be allocated from each cluster/group in the entire collection to construct a core collection on the basis of cluster/group size. The following strategies are implemented.

- Constant
- Proportional
- Logarithmic
- Square root

The different methods to determine the number of entries from each group or clusters implemented in `allocate.basic` are as follows.

Usage

```
allocate.basic(
  data,
  names,
  group,
  method = c("const", "prop", "log", "sqrt"),
  log.base = exp(1),
  size
)
```

Arguments

`data`	The data as a data frame object. The data frame should possess one row per individual and columns with the individual names and multiple trait/character data.
`names`	Name of column with the accession names as a character string.
`group`	Name of column with the accession group/cluster names as a character string.
`method`	The allocation method. Either `"const"` for constant or `"prop"` for proportional or `"log"` for logarithmic or `"sqrt"` for square root allocation.
`log.base`	The logarithm base to be used for logarithmic method of sampling. Default is `exp(1)`.
`size`	The desired core set size proportion.

Details

These are different methods which estimate the number of entries only on the basis of total number of entries in each cluster/group.

Brown (1989) proposed the constant (C), proportional (P) and logarithmic (L) methods and later a similar square root method was proposed by Huaman et al. (1999).

Constant method: From an entire collection of size N, to construct a core set of sample size n, the number of entries to be selected from the ith group among $1 \cdots g$ groups (n_i) is estimated as below.

$$n_i = \frac{n}{g} \times N$$

Proportional method: Here the number of entries to be selected is proportional to the cluster/group size (N_i) as below.

$$n_i = n \times \frac{N_i}{\sum_{i=1}^{g} N_i}$$

$$n_i = n \times \frac{N_i}{N}$$

Logarithmic method: Here the number of entries to be selected is proportional to the logarithm of the cluster/group size (N_i) as below.

$$n_i = n \times \frac{\log(N_i)}{\sum_{i=1}^{g} \log(N_i)}$$

Square root method: Here the number of entries to be selected is proportional to the square root of the cluster/group size (N_i) as below.

$$n_i = n \times \frac{\sqrt{N_i}}{\sum_{i=1}^{g} \sqrt{N_i}}$$

Value

A named numeric vector specifying the number of entries to be selected from each cluster/group. The vector names correspond to the levels of the ""group" column, and values indicate the number of elements to be selected from each level.

References

Brown AHD (1989). "Core collections: A practical approach to genetic resources management." *Genome*, **31**(2), 818–824.

Huaman Z, Aguilar C, Ortiz R (1999). "Selecting a Peruvian sweetpotato core collection on the basis of morphological, eco-geographical, and disease and pest reaction data:." *Theoretical and Applied Genetics*, **98**(5), 840–844.

See Also

allocate.distance, allocate.diversity

Examples

```
# Get data
data("cassava_EC_gp")

set.seed(123)
cassava_EC_gp <- cassava_EC_gp[sample(1:nrow(cassava_EC_gp), 500), ]

data <- cassava_EC_gp

data <- cbind(genotypes = rownames(cassava_EC_gp), cassava_EC_gp)
row.names(data) <- NULL

# Constant allocation
const_out <-
  allocate.basic(data = data, names = "genotypes",
                 group = "Cluster", method = "const",
                 size = 0.2)
const_out

# Proportional allocation
prop_out <-
  allocate.basic(data = data, names = "genotypes",
                 group = "Cluster", method = "prop",
                 size = 0.2)
prop_out

# Logarithmic allocation
log_out <-
  allocate.basic(data = data, names = "genotypes",
                 group = "Cluster", method = "log",
                 size = 0.2)
log_out
```

```
# Square root allocation
sqrt_out <-
  allocate.basic(data = data, names = "genotypes",
                 group = "Cluster", method = "sqrt",
                 size = 0.2)
sqrt_out
```

`allocate.distance` *Allocation of Entries to be Selected from Clusters/Groups based on Distance-based Diversity Metrics for Core Collection Development*

Description

Estimate the number of entries to be allocated from each cluster/group in the entire collection to construct a core collection on the basis of different metrics computed from within cluster/group distances. The following strategies are implemented.

- Diversity (Distance based)
- Diversity (Distance based) & Proportional
- Diversity (Distance based) & Logarithmic
- Diversity (Distance based) & Square root

Usage

```
allocate.distance(
  data,
  names,
  group,
  dist.mat,
  method = c("dist", "dist.prop", "dist.log", "dist.sqrt"),
 metric = c("mean", "median", "max", "range", "mnnd", "mdc", "mdm", "mstl", "nclust"),
  clust.fun = NULL,
  log.base = exp(1),
  size
)
```

Arguments

`data`	The data as a data frame object. The data frame should possess one row per individual and columns with the individual names and multiple trait/character data.
`names`	Name of column with the accession names as a character string.
`group`	Name of column with the accession group/cluster names as a character string.
`dist.mat`	A precomputed distance matrix of distance measures between the accessions in `data`.

`method`	The allocation method. Either `"dist"` for constant or `"dist.prop"` for proportional or `"dist.log"` for logarithmic or `"dist.sqrt"` for square root allocation. See **Methods**.
`metric`	The metric to be computed from the distance matrix. Either `"mean"`, `"median"`, `"max"`, `"range"`, `"mnnd"`, `"mdc"`, `"mdm"`, `"mstl"`, or `"nclust"`. See **Metrics**.
`clust.fun`	A function to generate clusters from a distance matrix and return the number of clusters.
`log.base`	The logarithm base to be used for logarithmic method of sampling. Default is `exp(1)`.
`size`	The desired core set size proportion.

Value

A named numeric vector specifying the number of entries to be selected from each cluster/group. The vector names correspond to the levels of the `""group"` column, and values indicate the number of elements to be selected from each level.

Details

The number of entries to be chosen from each cluster is estimated either on the basis of diversity of entries within that cluster/group alone or in combination with the size of the cluster/group (See **Methods**).

The within-cluster/group diversity is estimated as several metrics from the within cluster/group genetic distances between accessions (See **Metrics**).

Franco et al. (2005) proposed a method based on mean Gower's distance (Gower 1971) which was also extended to other distance measure averages named D Allocation strategy (Franco et al. 2006). These methods were also combined with the proportional and logarithmic methods. For example, the GP and GL strategy of Bisht et al. (1999) and Mahajan et al. (1999) as well as the NY and LD allocation methods of Franco et al. (2005).

Methods

Diversity method: From an entire collection of size N, to construct a core set of sample size n, the number of entries to be selected from the ith group among $1 \cdots g$ groups (n_i) is estimated as below.

$$n_i = n \times \frac{D_i}{\sum_{i=1}^{g} D_i}$$

Where, D_i is a measure of the extent of diversity present in the ith cluster.

Diversity and proportional method: Here the number of entries to be selected is proportional to the diversity of the cluster/group (D_i) weighted by the the cluster/group size (N_i).

$$n_i = n \times \frac{N_i D_i}{\sum_{i=1}^{g} N_i D_i}$$

Diversity and logarithmic method: Here the number of entries to be selected is proportional to the diversity of the cluster/group (D_i) weighted by the logarithm of the cluster/group size (N_i).

$$n_i = n \times \frac{\log(N_i)D_i}{\sum_{i=1}^{g} \log(N_i)D_i}$$

Diversity and square root method: Here the number of entries to be selected is proportional to the diversity of the cluster/group (D_i) weighted by the square root of the cluster/group size (N_i).

$$n_i = n \times \frac{\sqrt{N_i}D_i}{\sum_{i=1}^{g} \sqrt{N_i}D_i}$$

Metrics

Summary/Decriptive statistics: These include mean, median, maximum and range of genetic distances between entries in a cluster.

Mean nearest-neighbour distance ($MNND$): It is the average, across all entries, of the distance to each entry's closest other entry ($d_{g_{min}}$), based on a genetic given distance matrix (Clark and Evans 1954).

For each entry, the nearest-neighbour distance ($d_{g_{min}}$) is the smallest non-zero distance with any other entry.

$$d_{g_{min}} = \min_{h \neq g} d_{gh}$$

The Mean nearest-neighbour distance ($MNND$) can then be computed as:

$$\text{MNND} = \frac{1}{G} \sum_{g=1}^{G} d_g$$

Where, (g) is the index of an entry in a genetic distance matrix, h is the index of all other genotypes and G is the total number of genotypes in a cluster/group.

Minimum spanning tree length ($MSTL$): It is defined as the sum of edge weights in the minimum spanning tree constructed from the genetic distance matrix of entries within a cluster/group. A minimum spanning tree (MST) connects all entries such that the total distance is minimized and no cycles are formed. It represents the most efficient way to connect all entries based on pairwise genetic distances (Gower and Ross 1969).

For genetic distance d_{gh} between entries g and h, the MST is a subset of edges that connects all G entries with exactly $G - 1$ edges and minimum total weight. The MST length ($MSTL$) can then be computed as:

$$\text{MSTL} = \sum_{(g,h) \in \mathcal{T}} d_{gh}$$

Where $\mathcal{T}$ denotes the set of edges in the MST.

Mean distance to centroid and median (MDC, MDM): These quantify the average dispersion of entries within a cluster/group relative to a central point in multivariate space derived from the genetic distance matrix.

The centroid represents the multivariate mean position of all entries in a cluster (Sokal and Sneath 1963; Sneath and Sokal 1973)., whereas the median (spatial median) provides a robust central location that is less influenced by extreme values (Bradley et al. 1999).

For d_{gC} and d_{gM} distances of entry g from the centroid C and median M, respectively. These measures are computed as:

$$\text{MDC} = \frac{1}{G}\sum_{g=1}^{G} d_{gC}$$

$$\text{MDM} = \frac{1}{G}\sum_{g=1}^{G} d_{gM}$$

Where G is the total number of entries in the cluster/group.

Number of clusters: (Diwan et al. 1994) proposed the number of clusters produced by a multivariate cluster analysis at a specific distance threshold as an estimate of the diversity.

References

Bisht IS, Mahajan RK, Gautam PL (1999). "Assessment of genetic diversity, stratification of germplasm accessions in diversity groups and sampling strategies for establishing a core collection of Indian sesame (*Sesamum indicum* L.)." *Plant Genetic Resources Newsletter*, **199 Supp.**, 35–46.

Bradley PS, Bennett KP, Mangasarian OL (1999). "Constrained k-means clustering." Technical Report MSR-TR-2000-65, Microsoft Research, Redmond, WA.

Clark PJ, Evans FC (1954). "Distance to nearest neighbor as a measure of spatial relationships in populations." *Ecology*, **35**(4), 445–453.

Diwan N, Bauchan GR, McIntosh MS (1994). "A core collection for the united states annual *Medicago* germplasm collection." *Crop Science*, **34**(1), cropsci1994.0011183X003400010051x.

Franco J, Crossa J, Taba S, Shands H (2005). "A sampling strategy for conserving genetic diversity when forming core subsets." *Crop Science*, **45**(3), 1035–1044.

Franco J, Crossa J, Warburton ML, Taba S (2006). "Sampling strategies for conserving maize diversity when forming core subsets using genetic markers." *Crop Science*, **46**(2), 854–864.

Gower JC (1971). "A general coefficient of similarity and some of its properties." *Biometrics*, **27**(4), 857–871.

Gower JC, Ross GJS (1969). "Minimum spanning trees and single linkage cluster analysis." *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **18**(1), 54–64.

Mahajan RK, Bisht IS, Gautam PL (1999). "Sampling strategies for developing Indian sesame core collection." *Indian Journal of Plant Genetic Resources*, **12**(01), 1–9.

Sneath PHA, Sokal RR (1973). *Numerical Taxonomy: The Principles and Practice of Numerical Classification*, A Series of books in biology. W. H. Freeman, San Francisco. ISBN 978-0-7167-0697-7.

Sokal RR, Sneath PHA (1963). *Principles of numerical taxonomy*, A Series of books in biology. W. H. Freeman, San Francisco.

See Also

allocate.basic, allocate.diversity

Examples

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare example data
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

library(cluster)

# Get distance matrix
data("cassava_EC_gp")

set.seed(123)
cassava_EC_gp <- cassava_EC_gp[sample(1:nrow(cassava_EC_gp), 500), ]

quant <- c("NMSR", "TTRN", "TFWSR", "TTRW", "TFWSS", "TTSW", "TTPW",
           "AVPW", "ARSR", "SRDM")
qual <- c("CUAL", "LNGS", "PTLC", "DSTA", "LFRT", "LBTEF", "CBTR", "NMLB",
          "ANGB", "CUAL9M", "LVC9M", "TNPR9M", "PL9M", "STRP", "STRC",
          "PSTR")

data <- cassava_EC_gp

# Convert qualitative data columns to factor
data[, qual] <- lapply(data[, qual], as.factor)

# Standardise quantitative data column
data[, quant] <- lapply(data[, quant], function(x) {
  scale(x)[, 1]
})

# Get the Gower's distance matrix
dist_matrix <- daisy(x = data[, c(qual, quant)],
                     metric = "gower")

# Get data
data <- cassava_EC_gp
data <- cbind(genotypes = rownames(cassava_EC_gp), cassava_EC_gp)
row.names(data) <- NULL
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Custom clustering functions
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# UPGMA with hclust
clust_fun_upgma <- function(x) {
  # Tree
  tree_out <- hclust(x, method = "average")
  # Clusters
  cutree(tree_out, h = 0.2)
}

if (requireNamespace('fastcluster', quietly = TRUE)) {
  # Ward's minimum variance with fastcluster
  clust_fun_ward <- function(x) {
    # Tree
    tree_out <- fastcluster::hclust(x, method = "ward.D2")
    # Clusters
    cutree(tree_out, h = 0.2)
  }
}

if (requireNamespace('dbscan', quietly = TRUE)) {
  # Density-based clustering with dbscan
  clust_fun_dbscan <- function(x) {
    clust_out <- dbscan::dbscan(x, eps = 0.25)
    # remove noise: TODO
    setNames(clust_out$cluster, labels(x))
  }
}

if (requireNamespace('biotools', quietly = TRUE)) {
  # Tocher's sequential clustering
  clust_fun_tocher <- function(x) {
    clust_out <- biotools::tocher(x, algorithm = "sequential")
    setNames(clust_out$class, labels(x))
  }
}

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Diversity (Distance based) allocation
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Mean
dist_out_mean <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist",
                    dist.mat = dist_matrix, metric = "mean",
                    size = 0.2)
dist_out_mean

## Median
```

```
dist_out_median <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist",
                    dist.mat = dist_matrix, metric = "median",
                    size = 0.2)
dist_out_median

## Maximum
dist_out_max <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist",
                    dist.mat = dist_matrix, metric = "max",
                    size = 0.2)
dist_out_max

## Range
dist_out_range <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist",
                    dist.mat = dist_matrix, metric = "range",
                    size = 0.2)
dist_out_range

## Mean nearest-neighbour distance
dist_out_mnnd <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist",
                    dist.mat = dist_matrix, metric = "mnnd",
                    size = 0.2)
dist_out_mnnd

## Minimum spanning tree length
dist_out_mstl <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist",
                    dist.mat = dist_matrix, metric = "mstl",
                    size = 0.2)
dist_out_mstl


  ## Mean distance to centroid
  dist_out_mdc <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist",
                      dist.mat = dist_matrix, metric = "mdc",
                      size = 0.2)
  dist_out_mdc

  ## Mean distance to median
  dist_out_mdm <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist",
                      dist.mat = dist_matrix, metric = "mdm",
```

```
                      size = 0.2)
  dist_out_mdm


## Number of clusters

### UPGMA with hclust
dist_out_nclust1 <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist",
                    dist.mat = dist_matrix, metric = "nclust",
                    clust.fun = clust_fun_upgma,
                    size = 0.2)
dist_out_nclust1

# Ward's minimum variance with fastcluster
if (requireNamespace('fastcluster', quietly = TRUE)) {
  dist_out_nclust2 <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist",
                      dist.mat = dist_matrix, metric = "nclust",
                      clust.fun = clust_fun_ward,
                      size = 0.2)
  dist_out_nclust2
}


# Density-based clustering with dbscan
if (requireNamespace('dbscan', quietly = TRUE)) {
  dist_out_nclust3 <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist",
                      dist.mat = dist_matrix, metric = "nclust",
                      clust.fun = clust_fun_dbscan,
                      size = 0.2)
  dist_out_nclust3
}


  if (requireNamespace('biotools', quietly = TRUE)) {
    # Tocher's sequential clustering
    dist_out_nclust4 <-
      allocate.distance(data = data, names = "genotypes",
                        group = "Cluster", method = "dist",
                        dist.mat = dist_matrix, metric = "nclust",
                        clust.fun = clust_fun_tocher,
                        size = 0.2)
    dist_out_nclust4
  }


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Diversity (Distance based) & Proportional
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Mean
dist_prop_out_mean <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.prop",
                    dist.mat = dist_matrix, metric = "mean",
                    size = 0.2)
dist_prop_out_mean

## Median
dist_prop_out_median <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.prop",
                    dist.mat = dist_matrix, metric = "median",
                    size = 0.2)
dist_prop_out_median

## Maximum
dist_prop_out_max <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.prop",
                    dist.mat = dist_matrix, metric = "max",
                    size = 0.2)
dist_prop_out_max

## Range
dist_prop_out_range <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.prop",
                    dist.mat = dist_matrix, metric = "range",
                    size = 0.2)
dist_prop_out_range

## Mean nearest-neighbour distance
dist_prop_out_mnnd <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.prop",
                    dist.mat = dist_matrix, metric = "mnnd",
                    size = 0.2)
dist_prop_out_mnnd

## Minimum spanning tree length
dist_prop_out_mstl <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.prop",
                    dist.mat = dist_matrix, metric = "mstl",
                    size = 0.2)
dist_prop_out_mstl


  ## Mean distance to centroid
  dist_prop_out_mdc <-
```

```
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.prop",
                      dist.mat = dist_matrix, metric = "mdc",
                      size = 0.2)
  dist_prop_out_mdc

  ## Mean distance to median
  dist_prop_out_mdm <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.prop",
                      dist.mat = dist_matrix, metric = "mdm",
                      size = 0.2)
  dist_prop_out_mdm


## Number of clusters

### UPGMA with hclust
dist_prop_out_nclust1 <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.prop",
                    dist.mat = dist_matrix, metric = "nclust",
                    clust.fun = clust_fun_upgma,
                    size = 0.2)
dist_prop_out_nclust1

# Ward's minimum variance with fastcluster
if (requireNamespace('fastcluster', quietly = TRUE)) {
  dist_prop_out_nclust2 <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.prop",
                      dist.mat = dist_matrix, metric = "nclust",
                      clust.fun = clust_fun_ward,
                      size = 0.2)
  dist_prop_out_nclust2
}

# Density-based clustering with dbscan
if (requireNamespace('dbscan', quietly = TRUE)) {
  dist_prop_out_nclust3 <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.prop",
                      dist.mat = dist_matrix, metric = "nclust",
                      clust.fun = clust_fun_dbscan,
                      size = 0.2)
  dist_prop_out_nclust3
}


  if (requireNamespace('biotools', quietly = TRUE)) {
    # Tocher's sequential clustering
    dist_prop_out_nclust4 <-
      allocate.distance(data = data, names = "genotypes",
```

```
                        group = "Cluster", method = "dist.prop",
                        dist.mat = dist_matrix, metric = "nclust",
                        clust.fun = clust_fun_tocher,
                        size = 0.2)
    dist_prop_out_nclust4
  }


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Diversity (Distance based) & Logarithmic
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Mean
dist_log_out_mean <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.log",
                    dist.mat = dist_matrix, metric = "mean",
                    size = 0.2)
dist_log_out_mean

## Median
dist_log_out_median <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.log",
                    dist.mat = dist_matrix, metric = "median",
                    size = 0.2)
dist_log_out_median

## Maximum
dist_log_out_max <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.log",
                    dist.mat = dist_matrix, metric = "max",
                    size = 0.2)
dist_log_out_max

## Range
dist_log_out_range <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.log",
                    dist.mat = dist_matrix, metric = "range",
                    size = 0.2)
dist_log_out_range

## Mean nearest-neighbour distance
dist_log_out_mnnd <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.log",
                    dist.mat = dist_matrix, metric = "mnnd",
                    size = 0.2)
dist_log_out_mnnd

## Minimum spanning tree length
```

```
dist_log_out_mstl <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.log",
                    dist.mat = dist_matrix, metric = "mstl",
                    size = 0.2)
dist_log_out_mstl


  ## Mean distance to centroid
  dist_log_out_mdc <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.log",
                      dist.mat = dist_matrix, metric = "mdc",
                      size = 0.2)
  dist_log_out_mdc

  ## Mean distance to median
  dist_log_out_mdm <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.log",
                      dist.mat = dist_matrix, metric = "mdm",
                      size = 0.2)
  dist_log_out_mdm


## Number of clusters

### UPGMA with hclust
dist_log_out_nclust1 <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.log",
                    dist.mat = dist_matrix, metric = "nclust",
                    clust.fun = clust_fun_upgma,
                    size = 0.2)
dist_log_out_nclust1

# Ward's minimum variance with fastcluster
if (requireNamespace('fastcluster', quietly = TRUE)) {
  dist_log_out_nclust2 <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.log",
                      dist.mat = dist_matrix, metric = "nclust",
                      clust.fun = clust_fun_ward,
                      size = 0.2)
  dist_log_out_nclust2
}

if (requireNamespace('dbscan', quietly = TRUE)) {
  # Density-based clustering with dbscan
  dist_log_out_nclust3 <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.log",
                      dist.mat = dist_matrix, metric = "nclust",
```

```
                      clust.fun = clust_fun_dbscan,
                      size = 0.2)
  dist_log_out_nclust3
}


  if (requireNamespace('biotools', quietly = TRUE)) {
    # Tocher's sequential clustering
    dist_log_out_nclust4 <-
      allocate.distance(data = data, names = "genotypes",
                        group = "Cluster", method = "dist.log",
                        dist.mat = dist_matrix, metric = "nclust",
                        clust.fun = clust_fun_tocher,
                        size = 0.2)
    dist_log_out_nclust4
  }


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Diversity (Distance based) & Square root
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Mean
dist_sqrt_out_mean <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.sqrt",
                    dist.mat = dist_matrix, metric = "mean",
                    size = 0.2)
dist_sqrt_out_mean

## Median
dist_sqrt_out_median <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.sqrt",
                    dist.mat = dist_matrix, metric = "median",
                    size = 0.2)
dist_sqrt_out_median

## Maximum
dist_sqrt_out_max <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.sqrt",
                    dist.mat = dist_matrix, metric = "max",
                    size = 0.2)
dist_sqrt_out_max

## Range
dist_sqrt_out_range <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.sqrt",
                    dist.mat = dist_matrix, metric = "range",
                    size = 0.2)
dist_sqrt_out_range
```

```
## Mean nearest-neighbour distance
dist_sqrt_out_mnnd <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.sqrt",
                    dist.mat = dist_matrix, metric = "mnnd",
                    size = 0.2)
dist_sqrt_out_mnnd

## Minimum spanning tree length
dist_sqrt_out_mstl <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.sqrt",
                    dist.mat = dist_matrix, metric = "mstl",
                    size = 0.2)
dist_sqrt_out_mstl


  ## Mean distance to centroid
  dist_sqrt_out_mdc <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.sqrt",
                      dist.mat = dist_matrix, metric = "mdc",
                      size = 0.2)
  dist_sqrt_out_mdc

  ## Mean distance to median
  dist_sqrt_out_mdm <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.sqrt",
                      dist.mat = dist_matrix, metric = "mdm",
                      size = 0.2)
  dist_sqrt_out_mdm


## Number of clusters

### UPGMA with hclust
dist_sqrt_out_nclust1 <-
  allocate.distance(data = data, names = "genotypes",
                    group = "Cluster", method = "dist.sqrt",
                    dist.mat = dist_matrix, metric = "nclust",
                    clust.fun = clust_fun_upgma,
                    size = 0.2)
dist_sqrt_out_nclust1

# Ward's minimum variance with fastcluster
if (requireNamespace('fastcluster', quietly = TRUE)) {
  dist_sqrt_out_nclust2 <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.sqrt",
                      dist.mat = dist_matrix, metric = "nclust",
                      clust.fun = clust_fun_ward,
```

```
                      size = 0.2)
  dist_sqrt_out_nclust2
}

if (requireNamespace('dbscan', quietly = TRUE)) {
  # Density-based clustering with dbscan
  dist_sqrt_out_nclust3 <-
    allocate.distance(data = data, names = "genotypes",
                      group = "Cluster", method = "dist.sqrt",
                      dist.mat = dist_matrix, metric = "nclust",
                      clust.fun = clust_fun_dbscan,
                      size = 0.2)
  dist_sqrt_out_nclust3
}


  if (requireNamespace('biotools', quietly = TRUE)) {
    # Tocher's sequential clustering
    dist_sqrt_out_nclust4 <-
      allocate.distance(data = data, names = "genotypes",
                        group = "Cluster", method = "dist.sqrt",
                        dist.mat = dist_matrix, metric = "nclust",
                        clust.fun = clust_fun_tocher,
                        size = 0.2)
    dist_sqrt_out_nclust4
  }
```

`allocate.diversity` *Allocation of Entries to be Selected from Clusters/Groups based on Diversity Index Estimates for Core Collection Development*

Description

Estimate the number of entries to be allocated from each cluster/group in the entire collection to construct a core collection on the basis of different metrics computed from within cluster/group diversity index estimates. The following strategies are implemented.

- Diversity
- Diversity & Proportional
- Diversity & Logarithmic
- Diversity & Square root

Usage

```
allocate.diversity(
  data,
  names,
```

```
  group,
  qualitative,
  method = c("div", "div.prop", "div.sqrt", "div.log"),
  div.index = c("richness", "shannon", "simpson", "mcintosh"),
  shannon.base = exp(1),
  div.fun = NULL,
  log.base = exp(1),
  metric = c("pooled", "mean"),
  size
)
```

Arguments

data	The data as a data frame object. The data frame should possess one row per individual and columns with the individual names and multiple trait/character data.
names	Name of column with the accession names as a character string.
group	Name of column with the accession group/cluster names as a character string.
qualitative	Name of columns with the qualitative traits as a character vector.
method	The allocation method. Either "div" for constant or "div.prop" for proportional or "div.log" for logarithmic or "div.sqrt" for square root allocation.
div.index	The diversity index to be used to estimate within cluster/group diversity.
shannon.base	The logarithm base to be used for estimation of Shannon diversity index. Default is exp(1).
div.fun	A function to estimate diversity index from a factor vector of qualitative trait data.
log.base	The logarithm base to be used for logarithmic method of sampling. Default is exp(1).
metric	The metric to be computed from the diversity index. Either "pooled" or "mean".
size	The desired core set size proportion.

Value

A named numeric vector specifying the number of entries to be selected from each cluster/group. The vector names correspond to the levels of the ""group" column, and values indicate the number of elements to be selected from each level.

Details

The number of entries to be chosen from each cluster is estimated either on the basis of diversity of entries within that cluster/group alone or in combination with the size of the cluster/group (See Methods).

There are several methods proposed on the basis of diversity indices such as genetic multiplicity (G) dependent method based on the range of genetic diversity (Yonezawa et al. 1995), H strategy based on Nei's gene diversity (Nei 1973) and a method based on the pooled Shannon diversity index (Bisht et al. 1999; Mahajan et al. 1999). Similarly, measures such as expected proportion of heterozygous

loci per individual and effective number of alleles have also been employed as a diversity measure for determining sample size (Franco et al. 2006).

The within-cluster/group diversity is estimated as either pooled or mean value of cluster/group-wise diversity indices. The following diversity indices are implemented in this function.

- Shannon or Shannon-Weaver or Shannon-Wiener Diversity Index or Shannon entropy (H) (Shannon and Weaver 1949; Peet 1974)
- Simpson's Index of Diversity or Gini's Diversity Index or Gini-Simpson Index or Nei's Diversity Index or Nei's Variation Index (D) or Hurlbert's probability of interspecific encounter (PIE) (Gini 1912, 1912; Greenberg 1956; Berger and Parker 1970; Hurlbert 1971; Nei 1973; Peet 1974)
- McIntosh Diversity Index (D_{Mc}) (McIntosh 1967; Peet 1974)

Methods

Diversity method: From an entire collection of size N, to construct a core set of sample size n, the number of entries to be selected from the ith group among $1 \cdots g$ groups (n_i) is estimated as below.

$$n_i = n \times \frac{D_i}{\sum_{i=1}^{g} D_i}$$

Where, D_i is a measure of the extent of diversity present in the ith cluster.

Diversity and proportional method: Here the number of entries to be selected is proportional to the diversity of the cluster/group (D_i) weighted by the the cluster/group size (N_i).

$$n_i = n \times \frac{N_i D_i}{\sum_{i=1}^{g} N_i D_i}$$

Diversity and logarithmic method: Here the number of entries to be selected is proportional to the diversity of the cluster/group (D_i) weighted by the logarithm of the cluster/group size (N_i).

$$n_i = n \times \frac{\log(N_i) D_i}{\sum_{i=1}^{g} \log(N_i) D_i}$$

Diversity and square root method: Here the number of entries to be selected is proportional to the diversity of the cluster/group (D_i) weighted by the square root of the cluster/group size (N_i).

$$n_i = n \times \frac{\sqrt{N_i} D_i}{\sum_{i=1}^{g} \sqrt{N_i} D_i}$$

References

Berger WH, Parker FL (1970). "Diversity of planktonic foraminifera in deep-sea sediments." *Science*, **168**(3937), 1345–1347.

Bisht IS, Mahajan RK, Gautam PL (1999). "Assessment of genetic diversity, stratification of germplasm accessions in diversity groups and sampling strategies for establishing a core collection of Indian sesame (*Sesamum indicum* L.)." *Plant Genetic Resources Newsletter*, **199 Supp.**,

35–46.

Franco J, Crossa J, Warburton ML, Taba S (2006). "Sampling strategies for conserving maize diversity when forming core subsets using genetic markers." *Crop Science*, **46**(2), 854–864.

Gini C (1912). *Variabilita e Mutabilita. Contributo allo Studio delle Distribuzioni e delle Relazioni Statistiche. [Fasc. I.]*. Tipogr. di P. Cuppini, Bologna.

Gini C (1912). "Variabilita e mutabilita." In Pizetti E, Salvemini T (eds.), *Memorie di Metodologica Statistica*. Liberia Eredi Virgilio Veschi, Roma, Italy.

Greenberg JH (1956). "The measurement of linguistic diversity." *Language*, **32**(1), 109.

Hurlbert SH (1971). "The nonconcept of species diversity: A critique and alternative parameters." *Ecology*, **52**(4), 577–586.

Mahajan RK, Bisht IS, Gautam PL (1999). "Sampling strategies for developing Indian sesame core collection." *Indian Journal of Plant Genetic Resources*, **12**(01), 1–9.

McIntosh RP (1967). "An index of diversity and the relation of certain concepts to diversity." *Ecology*, **48**(3), 392–404.

Nei M (1973). "Analysis of gene diversity in subdivided populations." *Proceedings of the National Academy of Sciences*, **70**(12), 3321–3323.

Peet RK (1974). "The measurement of species diversity." *Annual Review of Ecology and Systematics*, **5**(1), 285–307.

Peet RK (1974). "The measurement of species diversity." *Annual Review of Ecology and Systematics*, **5**(1), 285–307.

Shannon CE, Weaver W (1949). *The Mathematical Theory of Communication*, number v. 2 in The Mathematical Theory of Communication. University of Illinois Press.

Yonezawa K, Nomura T, Morishima H (1995). "Sampling strategies for use in stratified germplasm collections." In Hodkin T, Brown ADH, van Hintum TJL, Morales EAV (eds.), *Core Collections of Plant Genetic Resources*, 35–53. John Wiley & Sons, New York. ISBN 0-471-95545-0.

See Also

`allocate.basic`, `allocate.distance`

Examples

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare example data
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Get data
```

```
data("cassava_EC_gp")

set.seed(123)
cassava_EC_gp <- cassava_EC_gp[sample(1:nrow(cassava_EC_gp), 500), ]

data <- cbind(genotypes = rownames(cassava_EC_gp), cassava_EC_gp)
row.names(data) <- NULL

# Column names of traits
quant <- c("NMSR", "TTRN", "TFWSR", "TTRW", "TFWSS", "TTSW", "TTPW",
           "AVPW", "ARSR", "SRDM")
qual <- c("CUAL", "LNGS", "PTLC", "DSTA", "LFRT", "LBTEF", "CBTR", "NMLB",
          "ANGB", "CUAL9M", "LVC9M", "TNPR9M", "PL9M", "STRP", "STRC",
          "PSTR")

# Convert qualitative data columns to factor
data[, qual] <- lapply(data[, qual], as.factor)

# Convert quantitative data columns to qualitative scores
quant_to_score5 <- function(x) {

  brks <- unique( quantile(x,
                           probs = seq(0, 1, 0.2),
                           na.rm = TRUE))
  cut(x, breaks = brks,
      include.lowest = TRUE,
      labels = seq_len(length(brks) - 1))
}

data[, quant] <- lapply(data[, quant], quant_to_score5)

traits <- c(quant, qual)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Custom diversity index functions
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

div_fun_brillouin <- function(x) {
  n <- tabulate(x)
  n <- n[n > 0]
  N <- sum(n)
  if (N <= 1) {
    return(0)
  }
  (lgamma(N + 1) - sum(lgamma(n + 1)))/N
}

div_fun_margalef <- function(x) {
  tab <- tabulate(x)
  tab <- tab[tab > 0]
  S <- length(tab)
  N <- length(x)
  if (N <= 1) {
```

```
    return(0)
  }
  (S - 1)/log(N)
}

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Diversity allocation
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Shannon-Weaver Diversity Index
div_out_shannon1 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div",
                     div.index = "shannon", metric = "pooled",
                     size = 0.2)
div_out_shannon1

div_out_shannon2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div",
                      div.index = "shannon", metric = "mean",
                      size = 0.2)
div_out_shannon2

##  Gini-Simpson Index
div_out_simpson1 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div",
                      div.index = "simpson", metric = "pooled",
                      size = 0.2)
div_out_simpson1

div_out_simpson2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div",
                      div.index = "simpson", metric = "mean",
                      size = 0.2)
div_out_simpson2

## McIntosh Diversity Index
div_out_mcintosh1 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div",
```

```
                      div.index = "mcintosh", metric = "pooled",
                      size = 0.2)
div_out_mcintosh1

div_out_mcintosh2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div",
                      div.index = "mcintosh", metric = "mean",
                      size = 0.2)
div_out_mcintosh2

## Richness
div_out_richness1 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div",
                      div.index = "richness", metric = "pooled",
                      size = 0.2)
div_out_richness1

div_out_richness2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div",
                      div.index = "richness", metric = "mean",
                      size = 0.2)
div_out_richness2

## Brillouin Diversity Index
div_out_brillouin1 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div",
                      div.fun = div_fun_brillouin, metric = "pooled",
                      size = 0.2)
div_out_brillouin1

div_out_brillouin2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div",
                      div.fun = div_fun_brillouin, metric = "mean",
                      size = 0.2)
div_out_brillouin2

## Margalef's richness Index
div_out_margalef1 <-
```

```
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div",
                     div.fun = div_fun_margalef, metric = "pooled",
                     size = 0.2)
div_out_margalef1

div_out_margalef2 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div",
                     div.fun = div_fun_margalef, metric = "mean",
                     size = 0.2)
div_out_margalef2


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Diversity allocation & Proportional
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Shannon-Weaver Diversity Index
dist_prop_out_shannon1 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.prop",
                     div.index = "shannon", metric = "pooled",
                     size = 0.2)
dist_prop_out_shannon1

dist_prop_out_shannon2 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.prop",
                     div.index = "shannon", metric = "mean",
                     size = 0.2)
dist_prop_out_shannon2

##  Gini-Simpson Index
dist_prop_out_simpson1 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.prop",
                     div.index = "simpson", metric = "pooled",
                     size = 0.2)
dist_prop_out_simpson1

dist_prop_out_simpson2 <-
  allocate.diversity(data = data, names = "genotypes",
```

```
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.prop",
                      div.index = "simpson", metric = "mean",
                      size = 0.2)
dist_prop_out_simpson2

## McIntosh Diversity Index
dist_prop_out_mcintosh1 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.prop",
                      div.index = "mcintosh", metric = "pooled",
                      size = 0.2)
dist_prop_out_mcintosh1

dist_prop_out_mcintosh2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.prop",
                      div.index = "mcintosh", metric = "mean",
                      size = 0.2)
dist_prop_out_mcintosh2

## Richness
div_out_richness1 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.log",
                      div.index = "richness", metric = "pooled",
                      size = 0.2)
div_out_richness1

div_out_richness2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.log",
                      div.index = "richness", metric = "mean",
                      size = 0.2)
div_out_richness2

## Brillouin Diversity Index
div_out_brillouin1 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.prop",
                      div.fun = div_fun_brillouin, metric = "pooled",
                      size = 0.2)
```

```
div_out_brillouin1

div_out_brillouin2 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.prop",
                     div.fun = div_fun_brillouin, metric = "mean",
                     size = 0.2)
div_out_brillouin2

## Margalef's richness Index
div_out_margalef1 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.prop",
                     div.fun = div_fun_margalef, metric = "pooled",
                     size = 0.2)
div_out_margalef1

div_out_margalef2 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.prop",
                     div.fun = div_fun_margalef, metric = "mean",
                     size = 0.2)
div_out_margalef2

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Diversity allocation & Logarithmic
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Shannon-Weaver Diversity Index
dist_log_out_shannon1 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.log",
                     div.index = "shannon", metric = "pooled",
                     size = 0.2)
dist_log_out_shannon1

dist_log_out_shannon2 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.log",
                     div.index = "shannon", metric = "mean",
                     size = 0.2)
dist_log_out_shannon2
```

```
##  Gini-Simpson Index
dist_log_out_simpson1 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.log",
                     div.index = "simpson", metric = "pooled",
                     size = 0.2)
dist_log_out_simpson1

dist_log_out_simpson2 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.log",
                     div.index = "simpson", metric = "mean",
                     size = 0.2)
dist_log_out_simpson2

## McIntosh Diversity Index
dist_log_out_mcintosh1 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.log",
                     div.index = "mcintosh", metric = "pooled",
                     size = 0.2)
dist_log_out_mcintosh1

dist_log_out_mcintosh2 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.log",
                     div.index = "mcintosh", metric = "mean",
                     size = 0.2)
dist_log_out_mcintosh2

## Richness
div_out_richness1 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.log",
                     div.index = "richness", metric = "pooled",
                     size = 0.2)
div_out_richness1

div_out_richness2 <-
  allocate.diversity(data = data, names = "genotypes",
                     group = "Cluster",
                     qualitative = traits,
                     method = "div.log",
```

```
                    div.index = "richness", metric = "mean",
                    size = 0.2)
div_out_richness2

## Brillouin Diversity Index
div_out_brillouin1 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.log",
                    div.fun = div_fun_brillouin, metric = "pooled",
                    size = 0.2)
div_out_brillouin1

div_out_brillouin2 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.log",
                    div.fun = div_fun_brillouin, metric = "mean",
                    size = 0.2)
div_out_brillouin2

## Margalef's richness Index
div_out_margalef1 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.log",
                    div.fun = div_fun_margalef, metric = "pooled",
                    size = 0.2)
div_out_margalef1

div_out_margalef2 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.log",
                    div.fun = div_fun_margalef, metric = "mean",
                    size = 0.2)
div_out_margalef2

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Diversity allocation & Square root
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## Shannon-Weaver Diversity Index
dist_sqrt_out_shannon1 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.sqrt",
                    div.index = "shannon", metric = "pooled",
```

```
                      size = 0.2)
dist_sqrt_out_shannon1

dist_sqrt_out_shannon2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.sqrt",
                      div.index = "shannon", metric = "mean",
                      size = 0.2)
dist_sqrt_out_shannon2

##  Gini-Simpson Index
dist_sqrt_out_simpson1 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.sqrt",
                      div.index = "simpson", metric = "pooled",
                      size = 0.2)
dist_sqrt_out_simpson1

dist_sqrt_out_simpson2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.sqrt",
                      div.index = "simpson", metric = "mean",
                      size = 0.2)
dist_sqrt_out_simpson2

## McIntosh Diversity Index
dist_sqrt_out_mcintosh1 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.sqrt",
                      div.index = "mcintosh", metric = "pooled",
                      size = 0.2)
dist_sqrt_out_mcintosh1

dist_sqrt_out_mcintosh2 <-
  allocate.diversity(data = data, names = "genotypes",
                      group = "Cluster",
                      qualitative = traits,
                      method = "div.sqrt",
                      div.index = "mcintosh", metric = "mean",
                      size = 0.2)
dist_sqrt_out_mcintosh2

## Richness
div_out_richness1 <-
  allocate.diversity(data = data, names = "genotypes",
```

```
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.sqrt",
                    div.index = "richness", metric = "pooled",
                    size = 0.2)
div_out_richness1

div_out_richness2 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.sqrt",
                    div.index = "richness", metric = "mean",
                    size = 0.2)
div_out_richness2

## Brillouin Diversity Index
div_out_brillouin1 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.sqrt",
                    div.fun = div_fun_brillouin, metric = "pooled",
                    size = 0.2)
div_out_brillouin1

div_out_brillouin2 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.sqrt",
                    div.fun = div_fun_brillouin, metric = "mean",
                    size = 0.2)
div_out_brillouin2

## Margalef's richness Index
div_out_margalef1 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.sqrt",
                    div.fun = div_fun_margalef, metric = "pooled",
                    size = 0.2)
div_out_margalef1

div_out_margalef2 <-
  allocate.diversity(data = data, names = "genotypes",
                    group = "Cluster",
                    qualitative = traits,
                    method = "div.sqrt",
                    div.fun = div_fun_margalef, metric = "mean",
                    size = 0.2)
div_out_margalef2
```

cassava_EC_gp *IITA Cassava Germplasm Data - Entire Collection*

Description

An example germplasm characterisation data of a subset of IITA Cassava collection (International Institute of Tropical Agriculture et al. 2019). Includes data on 26 (out of 62) descriptors for 1684 (out of 2170) accessions. The data has been partitioned into 6 clusters by hierarchial clustering. It is used to demonstrate the various functions of `SampleCore` package.

Usage

```
data(cassava_EC_gp)
```

Format

A data frame with 59 columns:

CUAL Colour of unexpanded apical leaves

LNGS Length of stipules

PTLC Petiole colour

DSTA Distribution of anthocyanin

LFRT Leaf retention

LBTEF Level of branching at the end of flowering

CBTR Colour of boiled tuberous root

NMLB Number of levels of branching

ANGB Angle of branching

CUAL9M Colours of unexpanded apical leaves at 9 months

LVC9M Leaf vein colour at 9 months

TNPR9M Total number of plants remaining per accession at 9 months

PL9M Petiole length at 9 months

STRP Storage root peduncle

STRC Storage root constrictions

PSTR Position of root

NMSR Number of storage root per plant

TTRN Total root number per plant

TFWSR Total fresh weight of storage root per plant

TTRW Total root weight per plant

TFWSS Total fresh weight of storage shoot per plant

TTSW Total shoot weight per plant

TTPW Total plant weight

AVPW Average plant weight

ARSR Amount of rotted storage root per plant

SRDM Storage root dry matter

Cluster The cluster to which the accessions belong identified by hierarchial clustering

Details

Further details on how the example dataset was built from the original data is available online. The details of how the clusters were identified are also available online.

References

International Institute of Tropical Agriculture, Benjamin F, Marimagne T (2019). "Cassava morphological characterization. Version 2018.1."

See Also

cassava_EC

Examples

```
data(cassava_EC_gp)
summary(cassava_EC_gp)

quant <- c("NMSR", "TTRN", "TFWSR", "TTRW", "TFWSS", "TTSW", "TTPW", "AVPW",
           "ARSR", "SRDM")
qual <- c("CUAL", "LNGS", "PTLC", "DSTA", "LFRT", "LBTEF", "CBTR", "NMLB",
          "ANGB", "CUAL9M", "LVC9M", "TNPR9M", "PL9M", "STRP", "STRC",
          "PSTR")

lapply(seq_along(cassava_EC_gp[, qual]),
       function(i) barplot(table(cassava_EC_gp[, qual][, i]),
                           xlab = names(cassava_EC_gp[, qual])[i]))

lapply(seq_along(cassava_EC_gp[, quant]),
       function(i) hist(table(cassava_EC_gp[, quant][, i]),
                        xlab = names(cassava_EC_gp[, quant])[i],
                        main = ""))
```

plot_dist *Plot a distance matrix as a 2D projection*

Description

Reduces a distance matrix to two dimensions using Classical MDS, Isotonic MDS, or t-SNE, and returns a ggplot2 scatter plot in which proximity reflects similarity. Points can optionally be highlighted or split into facet panels by group.

Usage

```
plot_dist(
  d,
  method = c("cmds", "isomds", "tsne"),
  highlight = NULL,
  gp = NULL,
  point.alpha = 0.8
)
```

Arguments

d	A distance matrix of class dist. Labels must be set (i.e. labels(d) must not be NULL). Duplicate labels are not permitted.
method	Character string specifying the dimensionality-reduction method. One of: "cmds" Classical (metric) Multidimensional Scaling via cmdscale. This is the default. "isomds" Non-metric (isotonic) MDS via isoMDS. Automatically falls back to "cmds" with a message when n < 3. "tsne" t-distributed Stochastic Neighbour Embedding via Rtsne. Perplexity is set automatically to min(30, floor((n - 1) / 3)).
highlight	Optional character vector of labels to highlight in the plot. Matching identifiers are plotted in **red**; all others in black. NULL (default) disables highlighting. Every value must be present in labels(d).
gp	Optional named character vector mapping labels to group names (names(gp) = labels, values = group names). When supplied, the plot is split into one facet panel per group via facet_wrap. The set of names must match labels(d) exactly. NULL (default) produces a single panel.
point.alpha	Alpha transparency value for points.

Value

A ggplot object. The plot can be further customised with standard **ggplot2** additions before printing or saving.

See Also

cmdscale, isoMDS, Rtsne, ggplot

Examples

```
# Basic usage with the built-in eurodist dataset
plot_dist(eurodist)

# Non-metric MDS with two highlighted cities
plot_dist(eurodist, method = "isomds",
          highlight = c("Madrid", "Rome"))

# Classical MDS split by a user-defined grouping
regions <-
  c(Athens = "South",  Barcelona = "South", Brussels = "North",
    Calais = "North",  Cherbourg = "North", Cologne = "North",
    Copenhagen = "North", Geneva = "South", Gibraltar = "South",
    Hamburg = "North", `Hook of Holland` = "North", Lisbon = "South",
    Lyons = "South",  Madrid = "South",  Marseilles = "South",
    Milan = "South",  Munich = "North",  Paris = "North",
    Rome = "South",   Stockholm = "North", Vienna = "North")

plot_dist(eurodist, method = "cmds", gp = regions,
          highlight = c("Madrid", "Cherbourg", "Rome", "Brussels"))
```

`select.distance` *Selection of Entries from Clusters/Groups on the basis of Genetic Distances*

Description

Select entries from cluster/groups in the entire collection by genetic distance based sampling according to allocation specified.

Usage

```
select.distance(
  data,
  names,
  group,
  alloc,
  dist.mat,
  always.selected = NULL,
 method = c("mean.medoid", "median.medoid", "nearest.centroid", "nearest.median",
   "mean.peripheral", "median.peripheral", "eccentricity", "farness.centrality",
   "kennard.stone", "duplex", "honigs", "farthest.point", "nearest.neighbour", "naes",
    "optim.medoid", "hclust.random", "hclust.medoid"),
```

```
  hclust.method = c("average", "single", "complete", "ward.D", "mcquitty", "median",
    "centroid", "ward.D2")
)
```

Arguments

data	The data as a data frame object. The data frame should possess one row per individual and columns with the individual names and multiple trait/character data.
names	Name of column with the accession names as a character string.
group	Name of column with the accession group/cluster names as a character string.
alloc	A named numeric vector specifying the number of entries to be selected. Names should correspond to the levels of the ""group" column, and values indicate the number of elements to be selected from each level.
dist.mat	A precomputed distance matrix of distance measures between the accessions in data.
always.selected	Names of accessions to be always included in the core set as a character vector.
method	The method for sampling accessions from each cluster/group. Either "mean.medoid", "median.medoid", "nearest.centroid", "nearest.median", "mean.peripheral", "median.peripheral", "eccentricity", "farness.centrality", "kennard.stone", "duplex", "honigs", "farthest.point", "nearest.neighbour", "naes", "optim.medoid", "hclust.random" or "hclust.medoid". See **Methods**.
hclust.method	The hierarchical clustering method to be used. Either "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).

Details

For each cluster/group, entries are selected by several methods from within-cluster/group genetic distances between accessions according to the allocation provided (See **Methods**).

Entries listed as always.selected are mandatorily included in the selection. Warnings are issued if requested allocation is smaller than the number of always-selected entries in a cluster/group and/or when the cluster/group does not contain enough remaining entries to fulfill the allocation.

Value

A named list where each element contains the selected entry identifiers for a cluster/group.

Methods

Centrality Based Methods:

Selects accessions that are most representative/closest to the cluster/group center.

Medoid-like Representative Sampling by Minimal Mean Distance: Selects medoid-like representatives as accessions with the smallest average distance to all others within the group (Kaufman and Rousseeuw 1987; Kaufman and Rousseeuw 1990).
For each accession g, the mean distance to all other accessions h is computed as:

$$\bar{d}_g = \frac{1}{G}\sum_{h=1}^{G} d_{gh}$$

Accessions are ranked by $\bar{d}_g$ in ascending order and the top n are selected.

Medoid-like Representative Sampling by Minimal Median Distance:
Selects medoid-like representatives as accessions with the smallest median distance to all others within the group. This method is less influenced by outliers (Kaufman and Rousseeuw 1987; Kaufman and Rousseeuw 1990).
For each accession g, the median distance to all other accessions h is computed as:

$$\tilde{d}_g = \text{median}_{h=1,\ldots,G}(d_{gh})$$

Accessions are ranked by $\tilde{d}_g$ in ascending order and the top n are selected.

Representative Sampling by Proximity to Group Centroid: Selects accessions closest to the group centroid in principal coordinate space, computed via multivariate dispersion analysis using `betadisper` (Anderson 2006; Anderson et al. 2006).
The distance of each accession g to the group centroid C in PCoA space is:

$$\delta_g = \|\mathbf{p}_g - \mathbf{c}\|$$

Where $\mathbf{p}_g$ is the PCoA coordinate vector of accession g and $\mathbf{c}$ is the group centroid. Accessions are ranked by δ_g in ascending order and the top n are selected.

Representative Sampling by Proximity to Group Spatial Median: Selects accessions closest to the group spatial median in principal coordinate space, computed via multivariate dispersion analysis using `betadisper` (O'Neill and Mathews 2000).
The distance of each accession g to the group spatial median M is:

$$\delta_g^* = \|\mathbf{p}_g - \mathbf{m}\|$$

where $\mathbf{m}$ is the spatial median of the group in PCoA space. Accessions are ranked by δ_g^* in ascending order and the top n are selected.

Peripheral/Extremity Based Methods:
Selects accessions that are most dissimilar from the rest in a cluster/group i.e. the accessions which are in the boundary or outliers.

Peripheral Sampling by Maximal Mean Distance: Selects the most peripheral accessions as those with the largest average distance to all others within the group (Kaufman and Rousseeuw 1987; Kaufman and Rousseeuw 1990).

$$\bar{d}_g = \frac{1}{G}\sum_{h=1}^{G} d_{gh}$$

Accessions are ranked by $\bar{d}_g$ in descending order and the top n are selected.

Peripheral Sampling by Maximal Median Distance: Selects the most peripheral accessions as those with the largest median distance to all others within the group (Kaufman and Rousseeuw 1987; Kaufman and Rousseeuw 1990).

$$\tilde{d}_g = \text{median}_{h=1,\ldots,G}(d_{gh})$$

Accessions are ranked by $\tilde{d}_g$ in descending order and the top n are selected.

Peripheral Sampling by Maximal Eccentricity: Selects accessions with the largest eccentricity — the maximum distance to any other accession in the group (Hage and Harary 1995).

$$e_g = \max_{h=1,\ldots,G} d_{gh}$$

Accessions are ranked by e_g in descending order and the top n are selected. Eccentricity captures the worst-case dissimilarity of an accession rather than its average behaviour.

Peripheral Sampling by Maximal Farness Centrality: Selects accessions with the greatest total distance to all others, i.e. those most remote from the rest of the group (Sabidussi 1966).

$$f_g = \sum_{h=1}^{G} d_{gh}$$

Accessions are ranked by f_g in descending order and the top n are selected. Farness centrality is proportional to $\bar{d}_g$ and differs from `mean.peripheral` only in that it uses the raw sum rather than the mean, producing identical rankings.

Space-Filling/Coverage Methods:

Select accessions that are spread maximally across the feature space in a cluster/group i.e. diversity sampling.

Space-Filling Sampling via the Kennard-Stone Algorithm: Selects n accessions that maximally and uniformly cover the distance space via the Kennard-Stone algorithm (Kennard and Stone 1969) (See `kenStone`).
Starting from the pair of accessions with the largest pairwise distance:

$$\{g_1, g_2\} = \arg\max_{g,h} d_{gh}$$

each subsequent accession g_k is selected by maximising its minimum distance to the already-selected set S:

$$g_k = \arg\max_{g \notin S} \min_{s \in S} d_{gs}$$

This greedy procedure ensures even space coverage without relying on cluster structure.

Space-Filling Sampling via the DUPLEX Algorithm: Extends the Kennard-Stone algorithm to simultaneously construct a model set and a test set with similar distributions (Kennard and Stone 1969; Snee 1977) (duplex). Accessions are selected using Mahalanobis distance:

$$d_M(g,h) = \sqrt{(\mathbf{x}_g - \mathbf{x}_h)^\top \Sigma^{-1} (\mathbf{x}_g - \mathbf{x}_h)}$$

where Σ is the covariance matrix. At each step, the pair maximising d_M is split alternately between model and test sets, ensuring both sets span the full feature space.

Space-Filling Sampling via the Honigs Algorithm: Selects n accessions sequentially by maximising dissimilarity to the already-selected set (Honigs et al. 1985) (honigs)
At each step k, the accession g_k maximising total distance to all previously selected accessions S is chosen:

$$g_k = \arg\max_{g \notin S} \sum_{s \in S} d_{gs}$$

This favours accessions that are collectively most dissimilar to the current selection, producing broad coverage of the distance space.

.

Space-Filling Sampling via Farthest-Point (Max-Min) Algorithm: Selects n accessions by iteratively maximising the minimum distance to the current selected set — also known as the max-min or farthest-point sampling algorithm (Gonzalez 1985; Dyer and Frieze 1985; Hochbaum and Shmoys 1985).

$$g_k = \arg\max_{g \notin S} \min_{s \in S} d_{gs}$$

This is equivalent to Kennard-Stone but without the symmetric initialisation step. It provides a deterministic, greedy approximation to the k-centre problem:

$$\min_{S \subset G,\, |S|=n} \max_{g \in G} \min_{s \in S} d_{gs}$$

Density Based Methods:

Select points based on local neighbourhood density.

Density-Based Sampling by Minimal Nearest-Neighbour Distance: Selects accessions residing in the densest regions of the distance space, identified as those with the smallest nearest-neighbour distance (Cover and Hart 1967; Fix and Hodges 1989).
For each accession g, the nearest-neighbour distance is:

$$\text{nn}_g = \min_{h \neq g} d_{gh}$$

Accessions are ranked by nn_g in ascending order and the top n are selected. Small nn_g indicates that g resides in a dense cluster; this method preferentially samples from high-density regions.

Cluster Based Methods:

These methods partition the cluster/group space into sub-clusters/groups, then samples from each one.

Globally Optimal Medoid Sampling via Partitioning Around Medoids (PAM):
Selects a set of n medoids that jointly minimise the total distance of every accession to its nearest medoid, via `pam`.
The objective function minimised is:

$$\min_{S \subset G,\, |S|=n} \sum_{g=1}^{G} \min_{s \in S} d_{gs}$$

Unlike `"mean.medoid"`, medoids are co-optimised as a set, ensuring they collectively represent the full distribution of the group rather than independently scoring each accession.

Cluster-Based Sampling via K-means (Naes Method): Partitions accessions into n clusters via k-means applied to the distance matrix (See `naes`), then selects the accession closest to each cluster centre as the representative (Naes 1987; Naes et al. 2017).
The k-means objective minimised is:

$$\min \sum_{k=1}^{n} \sum_{g \in C_k} d^2_{g,\mu_k}$$

where C_k is the k-th cluster and μ_k is its centre. One representative per cluster is returned, ensuring broad, partition-aware coverage.

Cluster-Based Sampling via Hierarchical Clustering with Random Selection: Partitions accessions into n clusters by cutting a hierarchical clustering dendrogram at height $k = n$, then randomly samples one accession from each cluster (Ward 1963; Li et al. 2002).
The dendrogram is built by agglomerative hierarchical clustering using the linkage criterion specified by `hclust`. For clusters $C_1, \ldots, C_n$, one accession is drawn uniformly at random from each:

$$g_k \sim \text{Uniform}(C_k), \quad k = 1, \ldots, n$$

This introduces stochasticity within a structured partition, balancing coverage with randomness.

Cluster-Based Sampling via Hierarchical Clustering with Medoid Selection: Partitions accessions into n clusters by cutting a hierarchical clustering dendrogram at height $k = n$, then selects the within-cluster medoid as the representative of each cluster (Kaufman and Rousseeuw 1987; Ward 1963).
For each cluster C_k, the medoid is the accession minimising total within-cluster distance:

$$g_k^* = \arg\min_{g \in C_k} \sum_{h \in C_k} d_{gh}$$

This combines the structured partitioning of hierarchical clustering with deterministic, centrality-based representative selection.

References

Anderson MJ (2006). "Distance-based tests for homogeneity of multivariate dispersions." *Biometrics*, **62**(1), 245–253.

Anderson MJ, Ellingsen KE, McArdle BH (2006). "Multivariate dispersion as a measure of beta diversity." *Ecology Letters*, **9**(6), 683–693.

Cover T, Hart P (1967). "Nearest neighbor pattern classification." *IEEE Transactions on Information Theory*, **13**(1), 21–27.

Dyer ME, Frieze AM (1985). "A simple heuristic for the *p*-centre problem." *Operations Research Letters*, **3**(6), 285–288.

Fix E, Hodges JL (1989). "Discriminatory analysis - Nonparametric discrimination: Consistency properties." *International Statistical Review / Revue Internationale de Statistique*, **57**(3), 238–247.

Gonzalez TF (1985). "Clustering to minimize the maximum intercluster distance." *Theoretical Computer Science*, **38**, 293–306.

Hage P, Harary F (1995). "Eccentricity and centrality in networks." *Social Networks*, **17**(1), 57–63.

Hochbaum DS, Shmoys DB (1985). "A best possible heuristic for the *K*-center problem." *Mathematics of Operations Research*, **10**(2), 180–184.

Honigs DE, Hieftje GM, Mark HL, Hirschfeld TB (1985). "Unique-sample selection via near-infrared spectral subtraction." *Analytical Chemistry*, **57**(12), 2299–2303.

Kaufman L, Rousseeuw PJ (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley Series in Probability and Statistics, 1 edition. Wiley. ISBN 978-0-471-87876-6 978-0-470-31680-1.

Kaufman P, Rousseeuw PJ (1987). "Clustering by means of medoids." In Dodge Y (ed.), *Proceedings of the Statistical Data Analysis Based on the L1 Norm Conference, Neuchatel, Switzerland*, volume 31, 405–416.

Kennard RW, Stone LA (1969). "Computer aided design of experiments." *Technometrics*, **11**(1), 137–148.

Li Z, Zhang H, Zeng Y, Yang Z, Shen S, Sun C, Wang X (2002). "Studies on sampling schemes for the establishment of core collection of rice landraces in Yunnan, China." *Genetic Resources and Crop Evolution*, **49**(1), 67–74.

Naes T (1987). "The design of calibration in near infra-red reflectance analysis by clustering." *Journal of Chemometrics*, **1**(2), 121–134.

Naes T, Isaksson T, Fearn T, Davies T (2017). *A User-Friendly Guide to Multivariate Calibration and Classification*, Second edition edition. IM Publications LLP, Chichester. ISBN 978-1-906715-25-0.

O'Neill ME, Mathews K (2000). "A weighted least squares approach to levene's test of homogeneity of variance." *Australian & New Zealand Journal of Statistics*, **42**(1), 81–100.

Sabidussi G (1966). "The centrality index of a graph." *Psychometrika*, **31**(4), 581–603.

Snee RD (1977). "Validation of regression models: Methods and examples." *Technometrics*, **19**(4), 415–428.

Ward JH (1963). "Hierarchical grouping to optimize an objective function." *Journal of the American Statistical Association*, **58**(301), 236–244.

See Also

`select.random`, `select.diversity`

Examples

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare example data
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

library(cluster)
library(ggplot2)
```

```
data(cassava_EC_gp)

set.seed(123)
data <- cassava_EC_gp[sample(1:nrow(cassava_EC_gp), 500), ]

quant <- c("NMSR", "TTRN", "TFWSR", "TTRW", "TFWSS", "TTSW", "TTPW", "AVPW",
           "ARSR", "SRDM")
qual <- c("CUAL", "LNGS", "PTLC", "DSTA", "LFRT", "LBTEF", "CBTR", "NMLB",
          "ANGB", "CUAL9M", "LVC9M", "TNPR9M", "PL9M", "STRP", "STRC",
          "PSTR")

data[, qual] <- lapply(data[, qual], as.factor)

# Get the Gower's distance matrix
dist_matrix <- daisy(x = data[, c(qual, quant)],
                     metric = "gower")


data <- cbind(genotypes = rownames(data), data)
row.names(data) <- NULL

# Prepare inputs
counts <- c(I = 16, II = 15, III = 9, IV = 18, V = 20, VI = 8)

mand_accns <-
  c("TMe-2018", "TMe-801", "TMe-3191", "TMe-1830", "TMe-1790")

gp_vec <- setNames(as.character(data[, "Cluster"]), data[, "genotypes"])


  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  # Fetch selected accessions by centrality based methods
  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  # Medoid-like Representative Sampling by Minimal Mean Distance
  sel_mean_medoid_out <-
    select.distance(data = data, names = "genotypes",
                    group = "Cluster", alloc = counts,
                    dist.mat = dist_matrix,
                    always.selected = mand_accns,
                    method = "mean.medoid")
  sel_mean_medoid_out

  plot_dist(d = dist_matrix, method = "isomds",
            gp = gp_vec,
            highlight =  unlist(sel_mean_medoid_out,
                                use.names = FALSE)) +
    labs(title = "mean.medoid")

  # Medoid-like Representative Sampling by Minimal Median Distance
  sel_median_medoid_out <-
    select.distance(data = data, names = "genotypes",
                    group = "Cluster", alloc = counts,
```

```
                dist.mat = dist_matrix,
                always.selected = mand_accns,
                method = "median.medoid")
sel_median_medoid_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_median_medoid_out,
                              use.names = FALSE)) +
  labs(title = "median.medoid")

# Representative Sampling by Proximity to Group Centroid
sel_group_centroid_out <-
  select.distance(data = data, names = "genotypes",
                group = "Cluster", alloc = counts,
                dist.mat = dist_matrix,
                always.selected = mand_accns,
                method = "nearest.centroid")
sel_group_centroid_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_group_centroid_out,
                              use.names = FALSE)) +
  labs(title = "nearest.centroid")

# Representative Sampling by Proximity to Group Spatial Median
sel_group_median_out <-
  select.distance(data = data, names = "genotypes",
                group = "Cluster", alloc = counts,
                dist.mat = dist_matrix,
                always.selected = mand_accns,
                method = "nearest.median")
sel_group_median_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_group_median_out,
                              use.names = FALSE)) +
  labs(title = "nearest.median")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Fetch selected accessions by peripheral/extremity based methods
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Peripheral Sampling by Maximal Mean Distance
sel_mean_peripheral_out <-
  select.distance(data = data, names = "genotypes",
                group = "Cluster", alloc = counts,
                dist.mat = dist_matrix,
                always.selected = mand_accns,
                method = "mean.peripheral")
sel_mean_peripheral_out
```

```
plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_mean_peripheral_out,
                              use.names = FALSE)) +
  labs(title = "mean.peripheral")

# Peripheral Sampling by Maximal Median Distance
sel_median_peripheral_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "median.peripheral")
sel_median_peripheral_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_median_peripheral_out,
                              use.names = FALSE)) +
  labs(title = "median.peripheral")

# Peripheral Sampling by Maximal Eccentricity
sel_eccentricity_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "eccentricity")
sel_eccentricity_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_eccentricity_out,
                              use.names = FALSE)) +
  labs(title = "eccentricity")

# Peripheral Sampling by Maximal Farness Centrality
sel_far_cent_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "farness.centrality")
sel_far_cent_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_far_cent_out,
                              use.names = FALSE)) +
  labs(title = "farness.centrality")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# Fetch selected accessions by space-Filling/coverage methods
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Space-Filling Sampling via the Kennard-Stone Algorithm
sel_ks_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "kennard.stone")
sel_ks_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_ks_out,
                              use.names = FALSE)) +
  labs(title = "kennard.stone")

# Space-Filling Sampling via the DUPLEX Algorithm
sel_duplex_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "duplex")
sel_duplex_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_duplex_out,
                              use.names = FALSE)) +
  labs(title = "duplex")

# Space-Filling Sampling via the Honigs Algorithm
sel_honigs_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "honigs")
sel_honigs_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_honigs_out,
                              use.names = FALSE)) +
  labs(title = "honigs")

# Space-Filling Sampling via Farthest-Point (Max-Min) Algorithm
sel_far_pt_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
```

```
                  always.selected = mand_accns,
                  method = "farthest.point")
sel_far_pt_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_far_pt_out,
                              use.names = FALSE)) +
  labs(title = "farthest.point")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Fetch selected accessions by density based methods
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Density-Based Sampling by Minimal Nearest-Neighbour Distance
sel_nn_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "nearest.neighbour")
sel_nn_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_nn_out,
                              use.names = FALSE)) +
  labs(title = "nearest.neighbour")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Fetch selected accessions by cluster based methods
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Globally Optimal Medoid Sampling via Partitioning Around Medoids (PAM)
sel_pam_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "optim.medoid")
sel_pam_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_pam_out,
                              use.names = FALSE)) +
  labs(title = "optim.medoid")

# Cluster-Based Sampling via K-means (Naes Method)
sel_naes_out <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
```

```
                  always.selected = mand_accns,
                  method = "naes")
sel_naes_out

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_naes_out,
                              use.names = FALSE)) +
  labs(title = "naes")

# Cluster-Based Sampling via Hierarchical Clustering with Random Selection

## UPGMA
sel_hclust_random_out1 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.random",
                  hclust.method = "average")
sel_hclust_random_out1

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_random_out1,
                              use.names = FALSE)) +
  labs(title = "hclust.random", subtitle = "average")

## Single-linkage
sel_hclust_random_out2 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.random",
                  hclust.method = "single")
sel_hclust_random_out2

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_random_out2,
                              use.names = FALSE)) +
  labs(title = "hclust.random", subtitle = "single")

## Complete-linkage
sel_hclust_random_out3 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.random",
                  hclust.method = "complete")
sel_hclust_random_out3
```

```
plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_random_out3,
                              use.names = FALSE)) +
  labs(title = "hclust.random", subtitle = "complete")

## Ward's D
sel_hclust_random_out4 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.random",
                  hclust.method = "ward.D")
sel_hclust_random_out4

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_random_out4,
                              use.names = FALSE)) +
  labs(title = "hclust.random", subtitle = "ward.D")

## WPGMA
sel_hclust_random_out5 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.random",
                  hclust.method = "mcquitty")
sel_hclust_random_out5

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_random_out5,
                              use.names = FALSE)) +
  labs(title = "hclust.random", subtitle = "mcquitty")

## WPGMC
sel_hclust_random_out6 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.random",
                  hclust.method = "median")
sel_hclust_random_out6

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_random_out6,
                              use.names = FALSE)) +
```

```
  labs(title = "hclust.random", subtitle = "median")

## UPGMC
sel_hclust_random_out7 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.random",
                  hclust.method = "centroid")
sel_hclust_random_out7

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_random_out7,
                              use.names = FALSE)) +
  labs(title = "hclust.random", subtitle = "centroid")

## Ward's D2
sel_hclust_random_out8 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.random",
                  hclust.method = "ward.D2")
sel_hclust_random_out8

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_random_out8,
                              use.names = FALSE)) +
  labs(title = "hclust.random", subtitle = "ward.D2")

# Cluster-Based Sampling via Hierarchical Clustering with Medoid Selection

## UPGMA
sel_hclust_medoid_out1 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.medoid",
                  hclust.method = "average")
sel_hclust_medoid_out1

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_medoid_out1,
                              use.names = FALSE)) +
  labs(title = "hclust.medoid", subtitle = "average")

## Single-linkage
```

```
sel_hclust_medoid_out2 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.medoid",
                  hclust.method = "single")
sel_hclust_medoid_out2

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_medoid_out2,
                              use.names = FALSE)) +
  labs(title = "hclust.medoid", subtitle = "single")

## Complete-linkage
sel_hclust_medoid_out3 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.medoid",
                  hclust.method = "complete")
sel_hclust_medoid_out3

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_medoid_out3,
                              use.names = FALSE)) +
  labs(title = "hclust.medoid", subtitle = "complete")

## Ward's D
sel_hclust_medoid_out4 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.medoid",
                  hclust.method = "ward.D")
sel_hclust_medoid_out4

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_medoid_out4,
                              use.names = FALSE)) +
  labs(title = "hclust.medoid", subtitle = "ward.D")

## WPGMA
sel_hclust_medoid_out5 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
```

```
                  method = "hclust.medoid",
                  hclust.method = "mcquitty")
sel_hclust_medoid_out5

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_medoid_out5,
                              use.names = FALSE)) +
  labs(title = "hclust.medoid", subtitle = "mcquitty")

## WPGMC
sel_hclust_medoid_out6 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.medoid",
                  hclust.method = "median")
sel_hclust_medoid_out6

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_medoid_out6,
                              use.names = FALSE)) +
  labs(title = "hclust.medoid", subtitle = "median")

## UPGMC
sel_hclust_medoid_out7 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.medoid",
                  hclust.method = "centroid")
sel_hclust_medoid_out7

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_hclust_medoid_out7,
                              use.names = FALSE)) +
  labs(title = "hclust.medoid", subtitle = "centroid")

## Ward's D2
sel_hclust_medoid_out8 <-
  select.distance(data = data, names = "genotypes",
                  group = "Cluster", alloc = counts,
                  dist.mat = dist_matrix,
                  always.selected = mand_accns,
                  method = "hclust.medoid",
                  hclust.method = "ward.D2")
sel_hclust_medoid_out8

plot_dist(d = dist_matrix, method = "isomds",
```

```
            gp = gp_vec,
            highlight =  unlist(sel_hclust_medoid_out8,
                                use.names = FALSE)) +
    labs(title = "hclust.medoid", subtitle = "ward.D2")
```

`select.diversity` *Selection of Entries from Clusters/Groups on the basis of Optimized Diversity*

Description

Select entries from cluster/groups in the entire collection which form a subset with the highest trait diversity according to a either pooled or mean diversity index estimate.

Usage

```
select.diversity(
  data,
  names,
  group,
  alloc,
  qualitative,
  always.selected = NULL,
  div.index = c("richness", "shannon", "simpson", "mcintosh"),
  shannon.base = exp(1),
  div.fun = NULL,
  metric = c("mean", "pooled"),
  search = c("random", "greedy"),
  local.search = c("best.improvement", "first.improvement"),
  n.iter = 1000,
  max.iter = 30
)
```

Arguments

`data`	The data as a data frame object. The data frame should possess one row per individual and columns with the individual names and multiple trait/character data.
`names`	Name of column with the accession names as a character string.
`group`	Name of column with the accession group/cluster names as a character string.
`alloc`	A named numeric vector specifying the number of entries to be selected. Names should correspond to the levels of the ""`group`" column, and values indicate the number of elements to be selected from each level.
`qualitative`	Name of columns with the qualitative traits as a character vector.

always.selected	Names of accessions to be always included in the core set as a character vector.
div.index	The diversity index to be used to estimate within cluster/group diversity.
shannon.base	The logarithm base to be used for estimation of Shannon diversity index. Default is exp(1).
div.fun	A function to estimate diversity index from a factor vector of qualitative trait data.
metric	The metric to be computed from the diversity index. Either "pooled" or "mean".
search	Character string specifying the search strategy used to find the subset with the highest diversity score. Either "random" (default) or "greedy" (See **Details**).
local.search	Character string specifying the local search strategy used in the 1-opt improvement phase of the greedy search (search = "greedy"). Either "best.improvement" (default) or "first.improvement". Ignored when search = "random".
n.iter	Integer specifying the number of random candidate subsets generated per group to optimze the diversity for random search (search = "random").
max.iter	The maximum number of 1-opt passes for greedy search (search = "greedy").

Details

To identify subsets with highest diversity estimates, the following strategies are available. These strategies are similar to the "Maximization" or M strategy of Schoen and Brown (1993).

Random search / Monte Carlo Method: For each cluster/group, multiple candidate subsets are sampled randomly and the subset with the highest trait diversity according to either pooled or mean diversity index estimate is retained. The quality of the solution improves with increasing n.iter but is not guaranteed to find the global optimum (Anatoly Zhigljavsky and Antanas Zilinskas 2008).

Greedy search with 1-opt: This method builds a solution incrementally by adding the accession that maximises the diversity score at each step, starting from the always.selected accessions (or a single randomly drawn accession when there are no accessions specified in always.selected) present in the particular cluster/group (Nemhauser et al. 1978; Fisher et al. 1978; Cormen et al. 2022). The 'greedy' solution is then refined by a 1-opt local search controlled by local.search and max.iter (Lin 1965). Greedy search is deterministic given a fixed always.selected set; when there are no accessions specified in always.selected present in the particular cluster/group results may vary across runs due to the random initialisation.

local.search = "best.improvement" scans all possible single swaps in each pass and applies the one yielding the greatest improvement before restarting. his guarantees the steepest ascent at each pass but requires evaluating all $k \times (n - k)$ swap pairs per pass, where k is the number of swappable accessions and $n - k$ is the size of the candidate pool (Papadimitriou and Steiglitz 1998).

local.search = "first.improvement" applies the first swap that improves the score and immediately restarts the search. This typically requires fewer score evaluations per pass and converges faster, but may find a different local optimum than "best.improvement" (Papadimitriou and Steiglitz 1998).

Both strategies terminate when no improving swap exists (local optimum) or when max.iter passes have been completed.

Entries listed as `always.selected` are mandatorily included in the selection. Warnings are issued if requested allocation is smaller than the number of always-selected entries in a cluster/group and/or when the cluster/group does not contain enough remaining entries to fulfill the allocation.

Value

A named list where each element contains the selected entry identifiers for a cluster/group.

References

Anatoly Zhigljavsky, Antanas Zilinskas (2008). *Stochastic Global Optimization*, volume 9 of *Springer Optimization and Its Applications*. Springer US, Boston, MA. ISBN 978-0-387-74022-5.

Cormen TH, Leiserson CE, Rivest RL, Stein C (2022). *Introduction to Algorithms*, 4 edition. MIT Press, Cambridge, MA, USA. ISBN 978-0-262-04630-5.

Fisher ML, Nemhauser GL, Wolsey LA (1978). "An analysis of approximations for maximizing submodular set functions-II." *Mathematical Programming Study*, **8**, 73–87.

Lin S (1965). "Computer solutions of the traveling salesman problem." *Bell System Technical Journal*, **44**(10), 2245–2269.

Nemhauser GL, Wolsey LA, Fisher ML (1978). "An analysis of approximations for maximizing submodular set functions-I." *Mathematical Programming*, **14**(1), 265–294.

Papadimitriou CH, Steiglitz K (1998). *Combinatorial optimization: Algorithms and complexity*. Dover Publications, Mineola, N.Y. ISBN 978-0-486-40258-1.

Schoen DJ, Brown AHD (1993). "Conservation of allelic richness in wild crop relatives is aided by assessment of genetic markers." *Proceedings of the National Academy of Sciences*, **90**(22), 10623–10627.

See Also

`select.random`, `select.distance`

Examples

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Prepare example data
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

library(cluster)
library(ggplot2)

data(cassava_EC_gp)

set.seed(123)
cassava_EC_gp <- cassava_EC_gp[sample(1:nrow(cassava_EC_gp), 500), ]
```

```
data <- cbind(genotypes = rownames(cassava_EC_gp), cassava_EC_gp)

quant <- c("NMSR", "TTRN", "TFWSR", "TTRW", "TFWSS", "TTSW", "TTPW", "AVPW",
           "ARSR", "SRDM")
qual <- c("CUAL", "LNGS", "PTLC", "DSTA", "LFRT", "LBTEF", "CBTR", "NMLB",
          "ANGB", "CUAL9M", "LVC9M", "TNPR9M", "PL9M", "STRP", "STRC",
          "PSTR")

# Convert qualitative data columns to factor
data[, qual] <- lapply(data[, qual], as.factor)

# Convert quantitative data columns to qualitative scores
quant_to_score5 <- function(x) {

  brks <- unique( quantile(x,
                           probs = seq(0, 1, 0.2),
                           na.rm = TRUE))
  cut(x, breaks = brks,
      include.lowest = TRUE,
      labels = seq_len(length(brks) - 1))
}

data[, quant] <- lapply(data[, quant], quant_to_score5)

traits <- c(quant, qual)

# Prepare inputs
counts <- c(I = 31, II = 31, III = 18, IV = 35, V = 40, VI = 17)

mand_accns <-
  c("TMe-2018", "TMe-801", "TMe-3191", "TMe-1830", "TMe-1790")

# Get distance matrix - Only for visualization

# Convert qualitative data columns to factor
cassava_EC_gp[, qual] <- lapply(cassava_EC_gp[, qual], as.factor)

# Standardise quantitative data column
cassava_EC_gp[, quant] <- lapply(cassava_EC_gp[, quant], function(x) {
  scale(x)[, 1]
})

gp_vec <- setNames(as.character(data[, "Cluster"]), data[, "genotypes"])

# Get the Gower's distance matrix
dist_matrix <- daisy(x = cassava_EC_gp[, c(qual, quant)],
                     metric = "gower")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Custom Diversity functions
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

div_fun_brillouin <- function(x) {
```

```
  n <- tabulate(x)
  n <- n[n > 0]
  N <- sum(n)
  if (N <= 1) {
    return(0)
  }
  (lgamma(N + 1) - sum(lgamma(n + 1)))/N
}

div_fun_margalef <- function(x) {
  tab <- tabulate(x)
  tab <- tab[tab > 0]
  S <- length(tab)
  N <- length(x)
  if (N <= 1) {
    return(0)
  }
  (S - 1)/log(N)
}


  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  # Random search
  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  # Mean richness
  randomsel_mean_richness <-
    select.diversity(data = data, names = "genotypes", group = "Cluster",
                     alloc = counts, qualitative = traits,
                     always.selected = mand_accns, div.index = "richness",
                     metric = "mean", search = "random", local.search = NULL,
                     n.iter = 50)
  randomsel_mean_richness

  plot_dist(d = dist_matrix, method = "isomds",
            gp = gp_vec,
            highlight =  unlist(randomsel_mean_richness,
                                use.names = FALSE)) +
    labs(title = "Random search", subtitle = "Mean richness")

  # Pooled richness
  randomsel_sum_richness <-
    select.diversity(data = data, names = "genotypes", group = "Cluster",
                     alloc = counts, qualitative = traits,
                     always.selected = mand_accns, div.index = "richness",
                     metric = "pooled", search = "random", local.search = NULL,
                     n.iter = 50)
  randomsel_sum_richness

  plot_dist(d = dist_matrix, method = "isomds",
            gp = gp_vec,
            highlight =  unlist(randomsel_sum_richness,
                                use.names = FALSE)) +
```

```
  labs(title = "Random search", subtitle = "Pooled richness")

# Mean Shannon-Weaver diversity index
randomsel_mean_shannon <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "shannon",
                   metric = "mean", search = "random", local.search = NULL,
                   n.iter = 50)
randomsel_mean_shannon

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_mean_shannon,
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Mean Shannon-Weaver diversity index")

# Pooled Shannon-Weaver diversity index
randomsel_sum_shannon <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "shannon",
                   metric = "pooled", search = "random", local.search = NULL,
                   n.iter = 50)
randomsel_sum_shannon

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_sum_shannon,
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Pooled Shannon-Weaver diversity index")

# Mean Gini-Simpson diversity index
randomsel_mean_simpson <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "simpson",
                   metric = "mean", search = "random", local.search = NULL,
                   n.iter = 50)
randomsel_mean_simpson

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_mean_simpson,
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Mean Gini-Simpson diversity index")

# Pooled Gini-Simpson diversity index
randomsel_sum_simpson <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
```

```
                  alloc = counts, qualitative = traits,
                  always.selected = mand_accns, div.index = "simpson",
                  metric = "pooled", search = "random", local.search = NULL,
                  n.iter = 50)
randomsel_sum_simpson

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_sum_simpson,
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Pooled Gini-Simpson diversity index")

# Mean McIntosh diversity index
randomsel_mean_mcintosh <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                  alloc = counts, qualitative = traits,
                  always.selected = mand_accns, div.index = "mcintosh",
                  metric = "pooled", search = "random", local.search = NULL,
                  n.iter = 50)
randomsel_mean_mcintosh

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_mean_mcintosh,
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Mean McIntosh diversity index")

# Pooled McIntosh diversity index
randomsel_sum_mcintosh <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                  alloc = counts, qualitative = traits,
                  always.selected = mand_accns, div.index = "mcintosh",
                  metric = "pooled", search = "random", local.search = NULL,
                  n.iter = 50)
randomsel_sum_mcintosh

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_sum_mcintosh,
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Pooled McIntosh diversity index")

# Mean Brillouin diversity index
randomsel_mean_brillouin <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                  alloc = counts, qualitative = traits,
                  always.selected = mand_accns, div.fun = div_fun_brillouin,
                  metric = "mean", search = "random", local.search = NULL,
                  n.iter = 50)
randomsel_mean_brillouin
```

```
plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_mean_brillouin,
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Mean Brillouin diversity index")

# Pooled Brillouin diversity index
randomsel_sum_brillouin <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.fun = div_fun_brillouin,
                   metric = "pooled", search = "random", local.search = NULL,
                   n.iter = 50)
randomsel_sum_brillouin

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_sum_brillouin,
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Pooled Brillouin diversity index")


# Mean Margalef's richness index
randomsel_mean_margalef <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.fun = div_fun_margalef,
                   metric = "mean", search = "random", local.search = NULL,
                   n.iter = 50)
randomsel_mean_margalef

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_mean_margalef,
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Mean Margalef's diversity index")

# Pooled Margalef's richness index
randomsel_sum_margalef <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.fun = div_fun_margalef,
                   metric = "pooled", search = "random", local.search = NULL,
                   n.iter = 50)
randomsel_sum_margalef

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(randomsel_sum_margalef,
```

```
                              use.names = FALSE)) +
  labs(title = "Random search",
       subtitle = "Pooled Margalef's diversity index")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Greedy search with 1-opt best improvement
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Mean richness
greedysel_best_mean_richness <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "richness",
                   metric = "mean", search = "greedy",
                   local.search = "best.improvement",max.iter = 3)
greedysel_best_mean_richness

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_mean_richness,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Mean richness")

# Pooled richness
greedysel_best_sum_richness <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "richness",
                   metric = "pooled", search = "greedy",
                   local.search = "best.improvement",max.iter = 3)
greedysel_best_sum_richness

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_sum_richness,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Pooled richness")

# Mean Shannon-Weaver diversity index
greedysel_best_mean_shannon <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "shannon",
                   metric = "mean", search = "greedy",
                   local.search = "best.improvement",max.iter = 3)
greedysel_best_mean_shannon

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_mean_shannon,
                              use.names = FALSE)) +
```

```
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Mean Shannon-Weaver diversity index")

# Pooled Shannon-Weaver diversity index
greedysel_best_sum_shannon <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "shannon",
                   metric = "pooled", search = "greedy",
                   local.search = "best.improvement",max.iter = 3)
greedysel_best_sum_shannon

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_sum_shannon,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Pooled Shannon-Weaver diversity index")

# Mean Gini-Simpson diversity index
greedysel_best_mean_simpson <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "simpson",
                   metric = "mean", search = "greedy",
                   local.search = "best.improvement",max.iter = 3)
greedysel_best_mean_simpson

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_mean_simpson,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Mean Gini-Simpson diversity index")

# Pooled Gini-Simpson diversity index
greedysel_best_sum_simpson <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "simpson",
                   metric = "pooled", search = "greedy",
                   local.search = "best.improvement",max.iter = 3)
greedysel_best_sum_simpson

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_sum_simpson,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Pooled Gini-Simpson diversity index")

# Mean McIntosh diversity index
greedysel_best_mean_mcintosh <-
```

```
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "mcintosh",
                   metric = "pooled", search = "greedy",
                   local.search = "best.improvement",max.iter = 3)
greedysel_best_mean_mcintosh

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_mean_mcintosh,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Mean McIntosh diversity index")

# Pooled McIntosh diversity index
greedysel_best_sum_mcintosh <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "mcintosh",
                   metric = "pooled", search = "greedy",
                   local.search = "best.improvement",max.iter = 3)
greedysel_best_sum_mcintosh

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_sum_mcintosh,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Pooled McIntosh diversity index")

# Mean Brillouin diversity index
greedysel_best_mean_brillouin <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns,
                   div.fun = div_fun_brillouin,
                   metric = "mean", search = "greedy",
                   local.search = "best.improvement",max.iter = 3)
greedysel_best_mean_brillouin

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_mean_brillouin,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Mean Brillouin diversity index")

# Pooled Brillouin diversity index
greedysel_best_sum_brillouin <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns,
                   div.fun = div_fun_brillouin,
```

```
                  metric = "pooled", search = "greedy",
                  local.search = "best.improvement",max.iter = 3)
greedysel_best_sum_brillouin

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_sum_brillouin,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Pooled Brillouin diversity index")

# Mean Margalef's richness index
greedysel_best_mean_margalef <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                  alloc = counts, qualitative = traits,
                  always.selected = mand_accns,
                  div.fun = div_fun_margalef,
                  metric = "mean", search = "greedy",
                  local.search = "best.improvement",max.iter = 3)
greedysel_best_mean_margalef

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_mean_margalef,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Mean Margalef's diversity index")

# Pooled Margalef's richness index
greedysel_best_sum_margalef <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                  alloc = counts, qualitative = traits,
                  always.selected = mand_accns,
                  div.fun = div_fun_margalef,
                  metric = "pooled", search = "greedy",
                  local.search = "best.improvement",max.iter = 3)
greedysel_best_sum_margalef

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_best_sum_margalef,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt best improvement",
       subtitle = "Pooled Margalef's diversity index")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Greedy search with 1-opt first improvement
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Mean richness
greedysel_first_mean_richness <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                  alloc = counts, qualitative = traits,
```

```
                  always.selected = mand_accns, div.index = "richness",
                  metric = "mean", search = "greedy",
                  local.search = "first.improvement",max.iter = 3)
greedysel_first_mean_richness

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_mean_richness,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Mean richness")

# Pooled richness
greedysel_first_sum_richness <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                  alloc = counts, qualitative = traits,
                  always.selected = mand_accns, div.index = "richness",
                  metric = "pooled", search = "greedy",
                  local.search = "first.improvement",max.iter = 3)
greedysel_first_sum_richness

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_sum_richness,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Pooled richness")

# Mean Shannon-Weaver diversity index
greedysel_first_mean_shannon <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                  alloc = counts, qualitative = traits,
                  always.selected = mand_accns, div.index = "shannon",
                  metric = "mean", search = "greedy",
                  local.search = "first.improvement",max.iter = 3)
greedysel_first_mean_shannon

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_mean_shannon,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Mean Shannon-Weaver diversity index")

# Pooled Shannon-Weaver diversity index
greedysel_first_sum_shannon <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                  alloc = counts, qualitative = traits,
                  always.selected = mand_accns, div.index = "shannon",
                  metric = "pooled", search = "greedy",
                  local.search = "first.improvement",max.iter = 3)
greedysel_first_sum_shannon
```

```
plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_sum_shannon,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Pooled Shannon-Weaver diversity index")

# Mean Gini-Simpson diversity index
greedysel_first_mean_simpson <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "simpson",
                   metric = "mean", search = "greedy",
                   local.search = "first.improvement",max.iter = 3)
greedysel_first_mean_simpson

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_mean_simpson,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Mean Gini-Simpson diversity index")

# Pooled Gini-Simpson diversity index
greedysel_first_sum_simpson <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "simpson",
                   metric = "pooled", search = "greedy",
                   local.search = "first.improvement",max.iter = 3)
greedysel_first_sum_simpson

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_sum_simpson,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Pooled Gini-Simpson diversity index")

# Mean McIntosh diversity index
greedysel_first_mean_mcintosh <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "mcintosh",
                   metric = "pooled", search = "greedy",
                   local.search = "first.improvement",max.iter = 3)
greedysel_first_mean_mcintosh

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_mean_mcintosh,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
```

```
       subtitle = "Mean McIntosh diversity index")

# Pooled McIntosh diversity index
greedysel_first_sum_mcintosh <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns, div.index = "mcintosh",
                   metric = "pooled", search = "greedy",
                   local.search = "first.improvement",max.iter = 3)
greedysel_first_sum_mcintosh

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_sum_mcintosh,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Pooled McIntosh diversity index")

# Mean Brillouin diversity index
greedysel_first_mean_brillouin <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns,
                   div.fun = div_fun_brillouin,
                   metric = "mean", search = "greedy",
                   local.search = "first.improvement",max.iter = 3)
greedysel_first_mean_brillouin

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_mean_brillouin,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Mean Brillouin diversity index")

# Pooled Brillouin diversity index
greedysel_first_sum_brillouin <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns,
                   div.fun = div_fun_brillouin,
                   metric = "pooled", search = "greedy",
                   local.search = "first.improvement",max.iter = 3)
greedysel_first_sum_brillouin

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_sum_brillouin,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Pooled Brillouin diversity index")

# Mean Margalef's richness index
```

```
greedysel_first_mean_margalef <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns,
                   div.fun = div_fun_margalef,
                   metric = "mean", search = "greedy",
                   local.search = "first.improvement",max.iter = 3)
greedysel_first_mean_margalef

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_mean_margalef,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Mean Margalef's richness index")

# Pooled Margalef's richness index
greedysel_first_sum_margalef <-
  select.diversity(data = data, names = "genotypes", group = "Cluster",
                   alloc = counts, qualitative = traits,
                   always.selected = mand_accns,
                   div.fun = div_fun_margalef,
                   metric = "pooled", search = "greedy",
                   local.search = "first.improvement",max.iter = 3)
greedysel_first_sum_margalef

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(greedysel_first_sum_margalef,
                              use.names = FALSE)) +
  labs(title = "Greed search | 1-opt first improvement",
       subtitle = "Pooled Margalef's richness index")
```

`select.random` *Selection of Entries from Clusters/Groups by Random Sampling*

Description

Select entries from cluster/groups in the entire collection by random sampling according to allocation specified.

Usage

```
select.random(data, names, group, alloc, always.selected = NULL)
```

Arguments

data	The data as a data frame object. The data frame should possess one row per individual and columns with the individual names and multiple trait/character data.
names	Name of column with the accession names as a character string.
group	Name of column with the accession group/cluster names as a character string.
alloc	A named numeric vector specifying the number of entries to be selected. Names should correspond to the levels of the ""group" column, and values indicate the number of elements to be selected from each level.
always.selected	Names of accessions to be always included in the core set as a character vector.

Details

For each cluster/group entries are selected randomly according to the allocation provided (Brown 1989; Brown and van Hintum 2000). Entries listed as always.selected are mandatorily included in the selection. Warnings are issued if requested allocation is smaller than the number of always-selected entries in a cluster/group and/or when the cluster/group does not contain enough remaining entries to fulfill the allocation.

Value

A named list where each element contains the selected entry identifiers for a cluster/group.

References

Brown AHD (1989). "Core collections: A practical approach to genetic resources management." *Genome*, **31**(2), 818–824.

Brown AHD, van Hintum TJL (2000). *Core Collections of Plant Genetic Resources*. Bioversity International. ISBN 92-9043-454-6.

See Also

select.distance, select.diversity

Examples

```
library(cluster)

# Get data
data(cassava_EC_gp)

set.seed(123)
cassava_EC_gp <- cassava_EC_gp[sample(1:nrow(cassava_EC_gp), 500), ]

data <- cbind(genotypes = rownames(cassava_EC_gp), cassava_EC_gp)
row.names(data) <- NULL
```

```
# Prepare inputs
counts <- c(I = 31, II = 31, III = 18, IV = 35, V = 40, VI = 17)

mand_accns <-
  c("TMe-2018", "TMe-801", "TMe-3191", "TMe-1830", "TMe-1790")

# Specify the seed
set.seed(123)

# Fetch selected accessions
sel_random_out <-
  select.random(data = data, names = "genotypes",
                group = "Cluster", alloc = counts,
                always.selected = mand_accns)

sel_random_out

# Get distance matrix - Only for visualization
quant <- c("NMSR", "TTRN", "TFWSR", "TTRW", "TFWSS", "TTSW", "TTPW",
           "AVPW", "ARSR", "SRDM")
qual <- c("CUAL", "LNGS", "PTLC", "DSTA", "LFRT", "LBTEF", "CBTR", "NMLB",
          "ANGB", "CUAL9M", "LVC9M", "TNPR9M", "PL9M", "STRP", "STRC",
          "PSTR")

# Convert qualitative data columns to factor
cassava_EC_gp[, qual] <- lapply(cassava_EC_gp[, qual], as.factor)

# Standardise quantitative data column
cassava_EC_gp[, quant] <- lapply(cassava_EC_gp[, quant], function(x) {
  scale(x)[, 1]
})

gp_vec <- setNames(as.character(data[, "Cluster"]), data[, "genotypes"])

# Get the Gower's distance matrix
dist_matrix <- daisy(x = cassava_EC_gp[, c(qual, quant)],
                     metric = "gower")

plot_dist(d = dist_matrix, method = "isomds",
          gp = gp_vec,
          highlight =  unlist(sel_random_out, use.names = FALSE))
```

Index