# Package ‘optpart’

January 19, 2020

**Version** 3.0-3

**Title** Optimal Partitioning of Similarity Relations

**Author** David W. Roberts <droberts@montana.edu>

**Maintainer** David W. Roberts <droberts@montana.edu>

**Depends** cluster, labdsv, MASS, plotrix

**Suggests** tree

**Description** Contains a set of algorithms for creating
partitions and coverings of objects largely based on operations
on (dis)similarity relations (or matrices). There are several
iterative re-assignment algorithms optimizing different
goodness-of-clustering criteria. In addition, there are
covering algorithms 'clique' which derives maximal cliques, and
'maxpact' which creates a covering of maximally compact sets.
Graphical analyses and conversion routines are also included.

**License** GPL (>= 2)

**URL** http://ecology.msu.montana.edu/labdsv/R

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2020-01-19 17:10:02 UTC

## R topics documented:

## `archi` *Archipelago Analysis*

### Description

Archipelago analysis finds connected clusters in a dissimilarity matrix. Samples in the same cluster are at most alpha dissimilar to at least one other sample in the cluster, and are more than alpha dissimilar to all samples in all other clusters. The solution is equivalent to slicing a nearest neighbor cluster analysis at alpha, but does not require (or produce) a hierarchical structure.

### Usage

```
archi(dist,alpha)
```

### Arguments

| | |
|---|---|
| `dist` | an object of class ‘dist’ from `dist`, `vegdist`, or `dsvdis` |
| `alpha` | the dissimilarity threshold to establish the relationship |

## Details

Archipelago analysis is a topological, as opposed to metric space, cluster routine that returns connected clusters. Every sample in a cluster is connected by a path with step lengths of at most alpha dissimilarity to every other sample in the cluster, and is more than alpha dissimilar to all other samples in all other clusters.

## Value

produces an object of class 'clustering', a list with a vector 'clustering' of cluster memberships

## Author(s)

David W. Roberts <droberts@montana.edu>

## Examples

```
data(shoshveg) # produces a vegetation dataframe
dis.bc <- dsvdis(shoshveg,'bray/curtis')
                # produces a Bray/Curtis dissimilarity matrix
arc.50 <- archi(dis.bc,0.5) # clusters at 0.5 dissimilarity
table(arc.50$clustering)
```

---

`bestfit` *Identify the Goodness-of-Fit of Cluster Members*

---

## Description

Sorts the members of clusters by maximum similarity goodness-of-fit

## Usage

```
bestfit(x,cluster)
```

## Arguments

| | |
|---|---|
| `x` | an object of class 'partana' or 'silhouette' |
| `cluster` | a specific cluster number |

## Details

Simply finds all members of a specific cluster and lists them in order of (1) mean similarity to their cluster (if x is an object of class 'partana') or silhouette width (if x is an object of class 'silhouette' as produced by functions in package 'cluster')

## Value

returns a data.frame with cluster members in column 'ID' and goodness-of-fit in column 'fit'

## Author(s)

David W. Roberts <droberts@montana.edu>

## References

http://ecology.msu.montana.edu/labdsv/

## See Also

typal

## Examples

```
data(shoshveg)                       # returns vegetation matrix
dis.bc <- dsvdis(shoshveg,'bray')    # Bray/Curtis dissimilarity matrix
opt.5 <- optpart(5,dis.bc)           # 5 cluster partition
print(class(opt.5))
fit <- bestfit(opt.5,1)              # goodness-of-fit for cluster 1
sil.5 <- silhouette(opt.5,dis.bc)    # calculate silhouette widths
fit2 <- bestfit(sil.5,1)             # goodness-of-fit for cluster 1
```

---

bestopt *Best Of Set Optimal Partitions From Random Starts*

---

## Description

Produces a specified number of optpart solutions from random starts, keeping the best result of the set

## Usage

```
bestopt(dist,numclu,numrep,maxitr=100)
```

## Arguments

| | |
|---|---|
| dist | an object of class 'dist' from dist, vegdist, or dsvdis, or a symmetric dissimilarity matrix |
| numclu | the number of clusters desired |
| numrep | the number of random starts requested |
| maxitr | the maximum number of iterations per replicate |

## Details

calls function optpart with an random initial assignment of items to clusters 'numitr' times, keeping the best result (highest within/among ratio observed). See optpart for more details.

## Value

an object of class partana, with components:

| | |
|---|---|
| `ptc` | the mean similarity of each item to each cluster |
| `ctc` | the mean similarity of each cluster to other clusters |
| `musubx` | the membership of each item in each cluster |
| `clustering` | the best 'crisp' partition from musubx |
| `ratio` | the within-cluster/among-cluster similarity ratio achieved at each iteration of the selected result. |

## Note

This is a simple wrapper function to automate independent random starts of function optpart.

## Author(s)

David W. Roberts `<droberts@montana.edu>`

## See Also

`optpart`, `partana`, `pam`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
x <- bestopt(dis.bc,5,10)
summary(x)
## Not run: plot(x)
```

---

`classmatch` *Classification Matching and Differencing*

---

## Description

Compares two classifications by cross-tabulating the assignment of objects to classes, and (optionally) produces a new classification to reflect the congruences and differences

## Usage

```
classmatch(x,y,type='full')
```

## Arguments

| | |
|---|---|
| `x` | an object of class 'clustering', 'partana', 'partition' or a vector identifying membership of objects in classes |
| `y` | an object of class 'clustering', 'partana', 'partition' or a vector identifying membership of objects in classes |
| `type` | a switch, either 'full' or 'direct', to control the parameters of the algorithm |

## Details

classmatch first calculates a cross-tabulation of the two classifications. Then, if ‘type=="full"’, the default, it finds all cases of agreement in order of number of objects. Objects are assigned to new clusters to reflect that order. It’s important to note that a single class may be partitioned into several new classes, and the the number of new classes produced may be higher than either of the classifications considered.

If ‘type=="direct"’ classmatch assumes a one-to-one relation between the two classifications compared. Classmatch finds the largest case of agreement, and assigns that match to class 1. It then zeros out the rows and columns corresponding to those classes, and iterates.

## Value

A list with components:

| | |
|---|---|
| `tab` | the cross-tabulation analyzed |
| `pairs` | the x and y values considered matched in order of solution |
| `partial` | a cumulative fraction of agreement as a function of number of clusters |
| `ord` | a table showing the order of new clusters |
| `combo` | a new vector of assignment of objects to clusters, only produced if ‘type=="full"’ |

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## References

http://ecology.msu.montana.edu/labdsv/R

## See Also

`partition`, `optpart`, `slice`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
opt.5 <- optpart(5,dis.bc)
pam.5 <- pam(dis.bc,5)
classmatch(opt.5,pam.5)
```

## clique *Maximal Clique Analysis*

### Description

Maximal clique analysis produces the set of maximal cliques of a dissimilarity or distance matrix. Maximal cliques are sets where every member of the set is <= alpha-dissimilar to every other member.

### Usage

```
clique(dist,alphac,minsize=1,mult=100)
## S3 method for class 'clique'
summary(object, ...)
## S3 method for class 'clique'
plot(x, panel = 'all', ...)
```

### Arguments

| | |
|---|---|
| `dist` | an object of class 'dist' from `dist`, `vegdist`, or `dsvdis` |
| `alphac` | the dissimilarity threshold to establish the relationship |
| `minsize` | the minimum size clique to list in the results |
| `mult` | scratch space multiplier to control stack size (see below) |
| `object` | an object of class 'clique' |
| `...` | ancillary arguments to `summary` or `plot` |
| `x` | an object of class 'clique' |
| `panel` | an integer switch to indicate which panel to plot |

### Details

Maximal clique analysis produces a covering, as opposed to a partition, i.e. objects can belong to more than one clique, and every object belongs to at least one clique. The maximal clique solution is solved for by symbolic computation, as opposed to numerical computation, and produces a unique solution. The number of cliques produced cannot be known beforehand, and can significantly exceed the number of objects. The 'mult' argument controls the size of the stack to hold intermediate terms in the equation as the solution proceeds. At each iteration, the algorithm simplifies the equation to the extent possible, and recovers space used to hold terms that have been eliminated. Nonetheless, it is possible for the equation to grow quite large at intermediate steps. The initial value of 'mult=100' sets the stack to 100 times the number of objects in the dissimilarity/distance matrix. If the memory allocated is exceeded, the output is set to NULL, and a message is printed to increase the 'mult' argument to a higher value.

## Value

produces a list with elements:

| | |
|---|---|
| `alphac` | the threshold value used to establish the cliques |
| `musubx` | a matrix of object membership in each of the maximal cliques |
| `member` | a list of members of each clique |

## Note

WARNING. The run time of maximal clique analysis is approximately $2^n + n$ for $n$ objects. The number of cliques generated, and the run time, is sensitive to 'alpha', as values of 'alpha' close to the mean dissimilarity of the matrix are likely to produce the most cliques and longest run time. A solution for 1200 objects once took approximately 20 CPU days on a SparcStation. The example shown below (100 objects) runs in a few seconds on a modern computer.

## Author(s)

David W. Roberts `<droberts@montana.edu>` `http://ecology.msu.montana.edu/labdsv/R`

## Examples

```
data(shoshveg) # produces a vegetation dataframe
dis.bc <- dsvdis(shoshveg,'bray/curtis')
    # produces a Bray/Curtis dissimilarity matrix
cli.50 <- clique(dis.bc,0.5) # clusters at 0.5 dissimilarity, likely
    # to run for a few seconds in most PCs
summary(cli.50)
```

---

`clique.test` *Clique Test*

---

## Description

The 'clique.test' function analyzes within-clique variability in attributes of objects other than those used to calculate the similarity relation. If the cliques exhibit a narrower range of values than expected at random it may be that the variable analyzed has an underlying role in determining the attributes on which the similarity is calculated.

## Usage

```
clique.test(cliq,env,minsize=2,plotit=FALSE)
```

## Arguments

| | |
|---|---|
| `cliq` | an object of class 'clique' |
| `env` | a continuous environmental variable to test |
| `minsize` | the minimum size clique to test for range |
| `plotit` | a switch to control plotting each clique individually |

## Value

Produces a vector of probabilities, one for each clique that expresses the probability of obtaining a range of 'env' as small as observed. Also produces a plot of the sorted probabilities on the current device.

## Note

The 'clique.test' function actually calls the `envrtest` function once for each clique and stores the associated probability as determined by `envrtest`

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## See Also

`clique`, `mss.test`, `envrtest`

## Examples

```
data(shoshveg)
data(shoshsite)
dis.bc <- dsvdis(shoshveg,'bray')
## Not run: cli.60 <- clique(dis.bc,0.60)             # will run for several
## Not run: print(clique.test(cli.60,shoshsite$swb))  # minutes
```

---

`clustering` *Clustering Object*

---

## Description

A clustering object is a list with a component called 'clustering' which is an integer vector of length n where n is the number of elements in a classification.

## Details

The clustering object is defined simply to allow a cleaner interface to functions in package 'cluster'.

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## References

`http://ecology.msu.montana.edu/labdsv/R`

## compare — *Compare Species Constancy for Specified Clusters*

### Description

Extracts specified columns from a constancy table (see `const`) and identifies species which occur in one of the two clusters (potential diagnostic species) or in both.

### Usage

```
compare(const,left,right,thresh=0.2)
```

### Arguments

| | |
|---|---|
| `const` | a constancy table produced by function `const` |
| `left` | a numeric cluster |
| `right` | a cluster number |
| `thresh` | a minimum differential abundance to list in the table |

### Details

compare extracts two columns (left and right) from a constancy table produced by `const`, and calculates the pairwise differences. Differences greater than the specified threshold appear in the set ‘left’; negative differences less then minus one times the threshold appear in the set ‘right’, and species which occur in both columns but with an absolute value of difference less than the threshold appear in set ‘both’.

### Value

a list with elements

| | |
|---|---|
| `left` | a data.frame of species diagnostic of set ‘left’ |
| `right` | a data.frame of species diagnostic of set ‘right’ |
| `both` | species occuring in both sets and diagnostic of neither |

### Author(s)

David W. Roberts `<droberts@montana.edu>`

### References

`http://ecology.msu.montana.edu/labdsv/R`

## Examples

```
data(shoshveg)        # returns vegetation data set
data(shoshsite)       # returns site data
elev.clust <- as.numeric(factor(cut(shoshsite$elevation,5)))
                      # 5 elevation bands
elev.const <- const(shoshveg,elev.clust)
compare(elev.const,1,2)   # identify diagnostic species
```

---

## `confus` *(Fuzzy) Confusion Matrix*

---

### Description

A confusion matrix is a cross-tabulation of actual class membership with memberships predicted by a discriminant function, classification tree, or other predictive model. A fuzzy confusion matrix is a confusion matrix that corrects for 'near misses' in prediction by comparing the similarity of the predicted type to the actual type and giving credit for the similarity.

### Usage

```
confus(clustering,model,diss=NULL)
```

### Arguments

| | |
|---|---|
| `clustering` | an object of class 'clustering' or a vector of (integer or factor) class membership values |
| `model` | a predictive model of class 'tree' or 'randomForest' |
| `diss` | optionally, a dissimilarity object of class 'dist' from 'dist', 'dsvdis', or 'vegdist' |

### Details

Cross-classifies each sample by actual class membership and predicted membership, computing overall accuracy, and the Kappa statistic of agreement. If a dissimilarity matrix is passed, calculates a fuzzy confusion matrix. In this case, correct predictions are assigned values of 1.0, and other predictions are given the value of the similarity of the two types an placed on the diagonal. The dissimilarity of the two types is added off the diagonal as fuzzy error.

### Value

produces a list with elements

| | |
|---|---|
| `matrix` | the (fuzzy) cross-tabulation matrix as a data.frame |
| `correct` | the fraction of (fuzzily) correctly predicted samples |
| `kappa` | the value of the Kappa statistic |
| `legend` | the text legend for the cross-tabulation matrix |

## Note

Confusion matrices are commonly computed in remote sensing applications, but are equally suited to the evaluation of any predictive methods of class membership or factors.

## Author(s)

David W. Roberts <droberts@montana.edu> http://ecology.msu.montana.edu/labdsv/R

## References

http://ecology.msu.montana.edu/labdsv/R

## Examples

```
data(shoshveg) # returns a data frame of vegetation data
data(shoshsite) # returns a data frame of site data
dis.bc <- dsvdis(shoshveg,'bray')
opt.5 <- optpart(5,dis.bc)
library(tree)
mod <- tree(factor(opt.5$clustering)~ elevation+slope+av,
            data=shoshsite)
confus(opt.5,mod)
confus(opt.5,mod,dis.bc)
```

---

`consider` *Recommendations for Possible Merging of Clusters*

---

## Description

Presents an ordered list of possible cluster combinations to consider for merging to simplify a classification.

## Usage

```
consider(part)
```

## Arguments

| | |
|---|---|
| `part` | an object of class 'partana' from functions `partana`, `optpart` or `bestopt` |

## Details

Simply sorts the cluster-to-cluster mean similarity matrix of a 'partana' object into a list sorted by mean similarity.

## Value

a data.frame with three elements:

| | |
|---|---|
| `row` | the current cluster number |
| `col` | the cluster to which it is most similar |
| `vals` | the mean similarity of the two clusters |

## Note

The listed combinations are not suggested to be optimal by any specific criterion. In fact, if the 'partana' object was generated by `optpart` or `bestopt` it is known that the suggested combinations are sub-optimal. Nevertheless, sometimes it is desirable to simplify a classification for other reasons.

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## References

`http://ecology.msu.montana.edu/labdsv/R`

## See Also

`optpart`, `bestopt`

## Examples

```
data(shoshveg)                     # returns a vegetation data.frame
dis.bc <- dsvdis(shoshveg,'bray') # calculates a Bray/Curtis
                                   # dissimilarity matrix
opt.5 <- optpart(5,dis.bc)         # generates a 5 cluster partition
consider(opt.5)                    # recommends possible clusters to merge
```

---

`disdiam` *Dissimilarity Diameters of a Partition*

---

## Description

Calculates the diameter (maximum within-cluster dissimilarity) of all clusters in a partition, as well as the average diameter across all clusters.

## Usage

```
disdiam(x,dist,digits)
## S3 method for class 'stride'
disdiam(x,dist,digits=3)
## S3 method for class 'disdiam'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | a vector of integers or an object of class 'clustering', 'partition', 'partana', or 'stride' |
| `dist` | an object of class 'dist' from `dist`, `dsvdis`, or `vegdist` |
| `digits` | the number of significant digits reported in the output |
| `...` | ancillary arguments to the print function |

## Details

disdiam is a cluster validation routine, and calculates the diameter (maximum within-cluster dissimilarity) of each cluster, as well as the average diameter of across all clusters of size greater than one.

## Value

A list with components:

| | |
|---|---|
| `diameters` | a data.frame with clusters as rows, and cluster ID, cluster size, and diameter as cols |
| `mean` | the weighted mean diameter of clusters of size greater than one. The mean is weighted for cluster size. |

## Author(s)

David W. Roberts `<droberts@montana.edu>`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray')
opt.5 <- optpart(5,dis.bc)
disdiam(opt.5,dis.bc)
```

---

`extract` *Extract A Specific Cluster Solution From A Stride*

---

## Description

Extracts a specified cluster solution from an object of class 'stride'. The desired solution is specified by the number of clusters.

## Usage

```
## S3 method for class 'stride'
extract(stride,k)
```

## Arguments

| | |
|---|---|
| `stride` | an object of class 'stride' from function `stride` |
| `k` | the number of clusters desired |

## Details

A stride object consists of a list with a data.frame of cluster solutions for varying numbers of clusters. Extract simply selects one column of this data.frame (specified by number of clusters desired, not column number) and returns that solution as an object of class 'clustering'.

## Value

an object of class 'clustering'.

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## Examples

```
data(shoshveg)                  # get vegetation data
dis.bc <- dsvdis(shoshveg,'bray')   # calculate dissimilarity
                                    #   matrix
avg.hcl <- hclust(dis.bc,'average') # average linkage cluster
                                    #   analysis
avg.2.10 <- stride(2:10,avg.hcl)    # compute stride
res <- extract(avg.2.10,8)          # extract 8-cluster solution
```

---

`flexbeta` *Calculate a Flexible-Beta Dendrogram*

---

## Description

Calculates Lance and Williams flexible-beta dendrogram with simplified argument

## Usage

```
flexbeta(dis,beta=-0.25,alpha=(1-beta)/2,gamma=0)
```

## Arguments

| | |
|---|---|
| `dis` | a distance or dissimilarity object of class 'dist' |
| `beta` | the Beta coefficient |
| `alpha` | the Alpha coefficients (assumed equal) |
| `gamma` | the gamma coefficient |

## Details

Calculates a flexible-beta dendrogram from a dissimilarity matrix specifying minimum parameters. The routine is simply a wrapper for the `agnes` function from package cluster with suitable arguments specified to achieve desired results. Ecologist in particular (but many others) find beta = -0.25 a good default.

## Value

An object of class 'hclust' for plotting and analysis like other hclust objects, as compared to objects of class 'agnes' as generated by the `agnes` function in package cluster.

## Author(s)

for the agnes function, Peter Rousseeuw for the original Fortran, Martin Maechler for the R code

for this function, David W. Roberts `<droberts@montana.edu>`

## References

Lance, G.N., and W.T. Williams (1966). A General Theory of Classifactory Sorting Strategies, I. Hierarchical Systems. Computer J. *9*, 373-380.

## See Also

`agnes`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray')
flexb <- flexbeta(dis.bc)
```

---

## `gensilwidth` *Generalized Silhouette Width*

---

## Description

Calculates mean cluster silhouette widths using a generalized mean.

## Usage

```
gensilwidth(clust, dist, p=1)
```

## Arguments

| | |
|---|---|
| `clust` | an integer vector of cluster memberships or a classification object of class 'clustering' |
| `dist` | an object of class 'dist' |
| `p` | the scaling parameter of the analysis |

## Details

gensilwidth calculates mean cluster silhouette widths using a generalized mean. The scaling parameter can be set between $[-\infty, \infty]$ where values less than one emphasize connectivity, and values greater than one emphasize compactedness. Individual sample unit silhouette widths are still calculated as $s_i = (b_i - a_i)/\max(b_i, a_i)$ where $a_i$ is the mean dissimilarity of a sample unit to the cluster to which it is assigned, and $b_i$ is the mean dissimilarity to the nearest neighbor cluster. Given $s_i$ for all members of a cluster, the generalized mean is calculated as

$$\bar{s} = \left( \frac{1}{n} \sum_{k=1}^{n} s_k^p \right)^{1/p}$$

Exceptions exist for specific values:

for p=0

$$s_i = \left( \prod_{k=1}^{n} s_k \right)^{1/n}$$

for p=$-\infty$

$$s_i = \min_{k=1}^{n} s_k$$

for p=$\infty$

$$s_i = \max_{k=1}^{n} s_k$$

$p = -1$ = harmonic mean, $p = 0$ = geometric mean, and $p = 1$ = arithmetic mean.

## Value

an object of class 'silhouette', a list with components

| | |
|---|---|
| `cluster` | the assigned cluster for each sample unit |
| `neighbor` | the identity of the nearest neighbor cluster for each sample unit |
| `sil_width` | the silhouette width for each sample unit |

## Author(s)

Attila Lengyel and Zoltan Botta-Dukat wrote the algorithm

David W. Roberts <`droberts@montana.edu`> `http://ecology.msu.montana.edu/labdsv/R`

## References

Lengyel, A. and Z. Botta-Dukat. 2019. Silhouette width using generalized mean: A flexible method for assessing clustering efficiency. Ecology and Evolution https://doi.org/10.1002/ece3.5774

## See Also

`silhouette`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray')
opt.5 <- optpart(5,dis.bc)
gensilwidth(opt.5,dis.bc)
```

---

## `lambda` *Goodman- Kruskal Lambda Index of Classification Association*

---

### Description

Compares two classifications by calculating the Goodman-Kruskal Index of association

### Usage

```
lambda(x,y,digits=5)
```

### Arguments

| | |
|---|---|
| `x` | an object of class ‘clustering’, ‘partana’, ‘partition’ or a vector identifying membership of objects in classes with names attribute |
| `y` | an object of class ‘clustering’, ‘partana’, ‘partition’ or a vector identifying membership of objects in classes with names attribute |
| `digits` | the number of digits of the statsitic to report |

### Details

lambda calculates the Goodman-Kruskal index of association:

$$\frac{\sum_i max_j(n_{ij}) + \sum_j max_i(n_{ij}) - max(n_{i,}) - max(n._j)}{2 * \sum_i \sum_j n_{ij} - max(n_{i,}) - max(n._j)}$$

### Value

Prints a cross-tabulated table and the lambda statistic, and (invisibly) returns the lambda statistic

### Author(s)

David W. Roberts `<droberts@montana.edu>`

### References

`http://ecology.msu.montana.edu/labdsv/R`

### See Also

`partition`, `optpart`, `slice`,`classmatch`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
opt.5 <- optpart(5,dis.bc)
pam.5 <- pam(dis.bc,5)
lambda(opt.5,pam.5)
```

---

## maxsimset *Maximally Similar Sets Analysis*

---

### Description

Maximally similar sets is an approach to deriving relatively homogeneous subsets of objects as determined by similarity of the composition of the objects. Maximally similar sets are a covering, as opposed to a partition, of objects. The sets so derived can be tested against random sets of the same size to determine whether a vector of independent data exhibits an improbably restricted distribution within the sets.

### Usage

```
maxsimset(dist,size=NULL,alphac=NULL,mean=FALSE)
mss.test(mss, env, panel = 'all', main = deparse(substitute(env)),
         ...)
## S3 method for class 'mss'
plot(x, ...)
## S3 method for class 'mss'
getsets(mss)
```

### Arguments

| | |
|---|---|
| `dist` | a dist object from `dist`, `dsvdis`, or `vegdist` |
| `size` | the size of desired sets |
| `alphac` | the alpha-cut to specify maximum dissimilarity for inclusion in a set |
| `mean` | if mean is FALSE (the default), the algorithm uses a furthest neighbor criterion; if mean is TRUE, it uses a mean similarity criterion |
| `mss` | an object of class ‘mss’ |
| `env` | a quantitative environmental variable for analysis |
| `main` | a title for the plot of mss.test |
| `panel` | an integer switch to indicate which panel to draw |
| `x` | an object of class ‘mss’ from maxsimset |
| `...` | ancillary arguments for ‘plot’ |

## Details

maxsimset starts with each sample as a seed, and adds the most similar plot to the set. Plots are added in turn to the set (up to the size specified, or to the maximum dissimilarity specified) in order of maximum similarity. If mean is FALSE, the sample most similar to set is the sample with the max-min similarity, that is, the sample whose minimum similarity to the set if highest, equivalent to furthest-neighbor or complete-linkage in cluster analysis. If mean is TRUE, the sample most similar to a set is the sample with highest mean similarity to the set. Once the sets are determined for each seed, the list is examined for duplicate sets, which are deleted, to return the list of unique sets.

If 'alphac' is specified, sets are grown to maximum size, or to maximum dissimilarity as specified by alphac, whichever is smaller.

The 'mss.test' function analyzes within-set variability in attributes of the objects other than those used to calculate the similarity relation. If maximally similar sets exhibit a narrower range of values than expected at random it may be that the variable analyzed has an underlying role in determining the attributes on which the similarity is calculated. The function 'plot' plots the sorted within-set range of values in red, and the sorted range of values of random sets of the same size in black. This followed by a boxplot of within-set values for the random replicates versus the observed sets, and calculates a Wilcoxon rank sum test of the difference.

'getsets' expands and pulls out the maximally similar sets as a list of logical membership vectors for use in other analyses.

## Value

an object of class 'mss', a list with elements:

| | |
|---|---|
| `musubx` | a matrix of sample membership in the sets where membership is given by the similarity with which a sample joined the set |
| `member` | a list of set members in the order they were added to the set |
| `numset` | the number of unique sets derived |
| `size` | the number of members in each set |
| `distname` | the name of the dissimilarity/distance object employed |

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## Examples

```
data(shoshveg)
data(shoshsite)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
mss.10 <- maxsimset(dis.bc,10)
## Not run: mss.test(mss.10,shoshsite$elevation)
      # plots graph and produces summary
```

## mergeclust *Merge Specified Clusters in a Classification*

### Description

Re-assigns members of one cluster to another specified cluster, reducing the number of clusters by one.

### Usage

```
mergeclust(clustering,from,to)
```

### Arguments

| | |
|---|---|
| `clustering` | a vector of (integer) cluster memberships, or an object of class'partition', 'partana', or 'clustering' |
| `from` | the cluster number to be vacated |
| `to` | the cluster to which members will be re-assigned |

### Details

The function simply renumbers members of one cluster with the number of another, but greatly simplifies managing the list objects class'partition', 'partana', or 'clustering' and simplifes the syntax.

### Value

A list object of class 'clustering' specifying cluster membership for every object.

### Author(s)

David W. Roberts <`droberts@montana.edu`>

### References

`http://ecology.msu.montana.edu/labdsv/R`

### See Also

`partition`, `partana`, and `clustering`

### Examples

```
data(shoshveg)                   # returns a vegetation data.frame
dis.bc <- dsvdis(shoshveg,'bray/curtis')   # returns a Bray/Curtis
                                 # dissimilarity matrix
opt.5 <- optpart(5,dis.bc)       # five cluster partition
opt.5a <- mergeclust(opt.5,5,4)  # reassigns member from cluster
                                 #   5 to 4
```

## murdoch *Indicator Species Analysis by Murdoch Preference Function*

### Description

Calculates the indicator value of species in a single cluster or environment type using the Murdoch Preference Function

### Usage

```
murdoch(comm,type,minval=0,minplt=10)
## S3 method for class 'murdoch'
summary(object,pval=0.05,digits=3,...)
## S3 method for class 'murdoch'
plot(x,axtype=1,pval=0.05,...)
## S3 method for class 'murdoch'
print(x,digits = 5, ...)
```

### Arguments

| | |
|---|---|
| `comm` | a matrix or data.frame of samples with species as columns and samples as rows |
| `type` | a logical vector with values of TRUE for samples in a specific cluster or type |
| `minval` | a threshold minimum abundance value to count as a presence |
| `minplt` | the minimum number of presences to include a species in the calculation |
| `object` | and object of class 'murdoch' |
| `pval` | the maximum probability to include a species in the summary table |
| `digits` | the number of digits to report |
| `...` | ancillary arguments to maintain compatibility with the generic summary function |
| `x` | an object of class 'murdoch' |
| `axtype` | a switch to control scaling of the x axis in the plot. 1=number of plots in the data set, other = number of presences in the type |

### Details

Calculates the indicator value of species for a specific type using the modified Murdoch statistic:

$$log((p/a) * (n - p_i)/n_i)$$

where: $p$ = number of samples where species is present, $a$ = number of samples where species is absent, $n$ = total number of samples (p+a), $p_i$ = number of samples in type i where species is present, $n_i$ = number of samples in type i.

Probabilities are based on the hypergeometric distribution calculation of having as many or more presences in a type as observed.

## Value

a list object of class 'murdoch' with components:

| | |
|---|---|
| `minplt` | the minimum number of presences to be included |
| `nplots` | the number of plots a species occurs in |
| `type` | the plot membership vector for the type |
| `pres` | the number of presences for species in the type |
| `abs` | the number of absences of species in the type |
| `murdoch` | the Murdoch value for species in the type |
| `pval` | the probability of getting such a high murdoch value |

## Note

Indicator value analysis is a set of techniques designed to identify species of special interest in clusters or types. The most widely used indicator species analysis was proposed by Dufrene and Legendre (1997), and is included in package 'labdsv' as `indval`. murdoch differs significantly from indval in assumption and objective, seeking to identify species that have improbable occurrences in types, regardless of their relative frequency in the type

## Author(s)

David W. Roberts with help from Ken Aho <droberts@montana.edu> http://ecology.msu.montana.edu/labdsv/R

## See Also

`indval`, `tabdev`

## Examples

```
data(shoshveg)                            # returns a vegetation dataframe
dis.bc <- dsvdis(shoshveg,'bray/curtis') # returns a dissimilarity
                                          #   matrix
opt.5 <- optpart(5,dis.bc)
plot(murdoch(shoshveg,opt.5$clustering==1))
```

---

`neighbor` *Neighbor Analysis of Partitions*

---

## Description

Calculates the nearest neighbor (least dissimilar cluster) for each item in partition to identify the topology of the partition.

## Usage

```
neighbor(x,all=FALSE)
```

### Arguments

| | |
|---|---|
| x | an object of class 'pam' or class 'partana' |
| all | a logical switch to control which items are included in the calculation |

### Details

Each item in a partition has membership in a cluster. The nearest neighbor of an item is the cluster to which the item is least dissimilar, other than the one to which it it belongs. If 'all' is TRUE, then every item is included in the analysis. If 'all' is FALSE, only 'misfits' are included in the calculation. If the first argument is an object of class 'pam', then a misfit is an item with a negative silhouette width (see `silhouette`). If the first argument is an object of class 'partana', a misfit is an item with lower mean dissimilarity to another cluster than to the one to which it belongs.

### Value

A table with clusters as rows, and neighbors as columns.

### Author(s)

David W. Roberts `<droberts@montana.edu>`

### Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray')
opt.5 <- optpart(5,dis.bc)
neighbor(opt.5,all=TRUE)
```

## optimclass — *Optimum Classification by Counts of Indicator Species*

### Description

Calculates the number of indicator species/cluster across a range of partitions

### Usage

```
optimclass(comm, stride, pval = 0.01, counts = 2)
```

### Arguments

| | |
|---|---|
| comm | a community matrix with sample units as rows and species as columns |
| stride | an object of class 'stride'from function `stride` |
| pval | the minimum probability for inclusion in the list of indicators |
| counts | the minimum number of clusters for inclusion in the list |

## Details

Calculates the number of indicator species/cluster and the number of clusters with at least ‘counts’ indicators, using the $\phi$ index to identify indicators with probabilities less than or equal to ‘pval’. Arguably the optimal partition is the one with the most indicator species and the most clusters with adequate indicators.

## Value

A data.frame of

| | |
|---|---|
| `clusters` | number of clusters |
| `sig.spc` | the number of species with significant indicator value |
| `sig.clust` | the number of clusters with at least ‘counts’ indicator species |

## Note

The concept and first implementation were by Tichy in software package ‘Juice’, and this is a simple port of the algorithm to R.

## Author(s)

Lubomir Tichy wrote the original algorithm

David W. Roberts `<droberts@montana.edu>`

## References

Tichy, L., M. Chytry, M. Hajek, S. Talbot, and Z. Botta-Dukat. 2010. OptimClass: Using species-to-cluster fidelity to determine the optimal partition in classification of ecological communities. J. Veg. Sci. 21:287-299.

## See Also

`indval`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray')
opt.2.10 <- stride(2:20,dis.bc)
## Not run: optimclass(shoshveg,opt.2.10)
```

## optindval — *Optimizing Classification by Maximizing Dufrene and Legendre's Indicator Value*

### Description

optindval is a iterative re-assignment classification algorithm that assigns samples to clusters to maximize the sum of indicator values.

### Usage

```
optindval(comm,clustering,maxitr=100,minsiz=5)
```

### Arguments

| | |
|---|---|
| `comm` | a vegetation or other taxon table with samples as rows and taxa as columns |
| `clustering` | an index of cluster membership for each sample. May be either a numeric vector of length equal to the number of samples, or an object that inherits from class 'cluster' |
| `maxitr` | the maximum number of iterations to attempt |
| `minsiz` | the minimum size cluster to consider reassigning a sample out of |

### Details

Iterative re-allocation algorithms temporarily re-assign each sample to each of the other possible clusters and calculate a goodness-of-clustering statistic for each re-assignment. The best of all possible re-assignments is then executed and the algorithm iterates until there are no more good re-assignments or the maximum number of iterations is reached. In optindval, the goodness-of-clustering statistic is the sum of Dufrene and Legendre indicator values

### Value

a list of class "optindval","clustering" with components:

| | |
|---|---|
| `numitr` | the number of iterations performed |
| `sums` | a vector of indicator value probability sums |
| `clustering` | the vector of cluster memberships (as integers) for each sample |

### Note

Like many iterative re-assignment algorithms, optindval is likely to be VERY slow from a random start or poor initial condition. optindval is maybe better used to polish existing classifications

### Author(s)

David W. Roberts `<droberts@montana.edu>`

## References

http://ecology.msu.montana.edu/labdsv/R

## See Also

`optpart`, `opttdev`, `optsil`

## Examples

```
data(shoshveg) # returns a data.frame of vegetation data called shoshveg
dis.bc <- dsvdis(shoshveg,'bray') # generate Bray/Curtis dissimilarity
                                  # matrix
opt.5 <- optpart(5,dis.bc) # generate 5-cluster optpart
## Not run: res <- optindval(shoshveg,opt.5) # polish the optpart result
## Not run: classmatch(opt.5,res) # see the plot re-assignments
```

---

`optpart` *Optimal Partitioning of Dissimilarity/Distance Matrices*

---

## Description

Optimal partitioning is an iterative re-allocation algorithm to maximize the ratio of within-cluster similarity/among-cluster similarity for a given number of clusters. Optpart can operate as either a crisp (classical) partitioning, or a fuzzy partitioning algorithm.

## Usage

```
optpart(x, dist, maxitr = 100, mininc = 0.001, maxdmu = 1)
```

## Arguments

| | |
|---|---|
| `x` | an integer, integer vector, factor vector, or objects of class ‘clustering’, ‘partana’, ‘partition’ or ‘stride’ |
| `dist` | a object of class ‘dist’ from `dist`, `dsvdis`, or `vegdist` |
| `maxitr` | the maximum number of iterations to perform |
| `mininc` | the minimum increment in the within/among similarity ratio to continue iterating |
| `maxdmu` | the ‘maximum delta mu’. If 1, a crisp (non-fuzzy) partition results. If (0,1) a fuzzy partition results. |

## Details

optpart produces a partition, or clustering, of items into clusters by iterative reallocation of items to clusters so as to maximize the within cluster/ among cluster similarity ratio. At each iteration optpart ranks all possible re-allocations of a sample from one cluster to another. The re-allocation that maximizes the change in the within-cluster/among-cluster ratio is performed. The next best reallocation is considered, and if it does not include any clusters already modified, it is also performed, as re-allocations of independent clusters are independent and additive in effect. When no further re-allocations can be performed in that iteration, the algorithm recalculates all possible re-allocations and iterates again. When no re-allocations exist that improve the within/among ratio greater than ‘mininc’, or the maximum number of iterations is reached, the algorithm stops.

optpart is designed to run from a random start or the levels of a factor, or preferably from existing initial partitions. Specifying a single integer gives the number of clusters desired using a random start. Specifying an integer vector gives the initial assignments of items to clusters. Initial assignments can also be extracted from a number of objects. Specific methods exist for objects of class ‘clustering’ from functions `slice` or `archi`, class ‘partana’ from function `partana`, class ‘stride’ from `stride`, or class ‘partition’ from functions `pam` or `diana`. optpart is deterministic from a given initial condition. To get good results from a random start, multiple random starts should be attempted, using function `bestopt`.

Optpart is an unweighted algorithm, i.e. each of the $(n^2 - n)/2$ pairwise distances or dissimilarities is included in the calculation of the ratio exactly once. Optpart somewhat penalizes small clusters, as small clusters contribute only $(n_i^2 - n_i)/2$ values to the numerator; the extreme case is that a cluster with a single member does not contribute anything to the numerator.

It is an interesting characteristic of optpart that no minimum cluster size is enforced, and it is common for partitions of a large number of clusters to contain null clusters, i.e. clusters with no members. This is not a bug or error, but rather an indication that a partition with a fewer number of clusters achieves a better within/among similarity ratio than does a larger number. It is also somewhat common that for solutions with a small or intermediate number of clusters, optpart places outliers in a small ‘trash’ cluster.

When optpart is run as a fuzzy partitioning algorithm, it often achieves a surprisingly low entropy, with many items assigned completely to a single cluster.

## Value

an object of class `partana`, a list with elements:

| | |
|---|---|
| `ptc` | a matrix of item mean similarity to each cluster |
| `ctc` | a matrix of mean cluster-to-cluster similarity |
| `musubx` | a matrix of membership of each item to each cluster. If `maxdmu` is 1, this will be a single 1 in the appropriate cluster and 0 in all others. If `maxdmu` is (0,1) then the musubx represent fuzzy memberships in each cluster. |
| `clustering` | a vector giving the cluster each item is assigned to. If optpart is run as a fuzzy partitioning, this is determined by the maximum membership observed. |
| `ratio` | the vector of within/among similarities achieved at each iteration. The final non-zero value is the final ratio achieved. |
| `numitr` | the number of iterations performed |
| `names` | the names of the items clustered |

## Author(s)

David W. Roberts <`droberts@montana.edu`> `http://ecology.msu.montana.edu/labdsv/R`

## See Also

`partana`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
opt.5 <- optpart(5,dis.bc)
summary(opt.5)
```

---

## optsil *Clustering by Optimizing Silhouette Widths*

---

### Description

Silhouette width is a measurement of the mean similarity of each object to the other objects in its cluster, compared to its mean similarity to the most similar cluster (see `silhouette`). Optsil is an iterative re-allocation algorithm to maximize the mean silhouette width of a clustering for a given number of clusters.

### Usage

```
optsil(x,dist,maxitr)
```

### Arguments

| | |
|---|---|
| `x` | an integer, a vector of integers, an object of class 'clustering', 'partana', 'partition', or 'stride' |
| `dist` | a object of class 'dist' from `dist`, `dsvdis`, or `vegdist` |
| `maxitr` | the maximum number of iterations to perform |

### Details

optsil produces a partition, or clustering, of items into clusters by iterative reallocation of items to clusters so as to maximize the mean silhouette width of the classification. At each iteration optsil ranks all possible re-allocations of a item from one cluster to another. The reallocation that maximizes the change in the mean silhouette width is performed. Because silhouette widths are not independent of clusters that are not modified, only a single reallocation can be preformed in a single iteration. When no further re-allocations result in an improvement, or the maximum number of iterations is achieved, the algorithm stops.

Optsil is an unweighted algorithm, i.e. each of the objects is included in the calculation exactly once.

Optsil can be extremely slow to converge, and is best used to 'polish' an existing partition or clusterings resulting from slicing an `hclust` or from functions `optpart`, `pam`, `diana` or other initial clusterings. It is possible to run optsil from a random start, but is EXTREMELY SLOW to converge, and should be done only with caution.

## Value

a list with elements:

| | |
|---|---|
| `clustering` | a vector of integers giving the cluster assignment for each object |
| `sils` | a vector of the silhouette widths achieved at each iteration |
| `numitr` | the number of iterations performed |

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## See Also

`optpart`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
opt.5 <- optpart(5,dis.bc)
sil.5 <- optsil(opt.5,dis.bc,100) # make take a few minutes
summary(silhouette(sil.5,dis.bc))
## Not run: plot(silhouette(sil.5,dis.bc))
```

---

`opttdev` *Optimizing Classification by Minimizing Table Deviance*

---

## Description

opttdev is a iterative re-assignment classification algorithm that assigns samples to clusters to minimize the total deviance of a table with respect to the row-wise relative abundance of the elements

## Usage

```
opttdev(comm,clustering,maxitr=100,minsiz=5)
```

## Arguments

| | |
|---|---|
| `comm` | a vegetation or other taxon table with samples as rows and taxa as columns |
| `clustering` | an index of cluster membership for each sample. May be either a numeric vector of length equal to the number of samples, or an object that inherits from class ‘cluster’ |
| `maxitr` | the maximum number of iterations to attempt |
| `minsiz` | the minimum size cluster to consider reassigning a sample out of |

## Details

Iterative re-allocation algorithms temporarily re-assign each sample to each of the other possible clusters and calculate a goodness-of-clustering statistic for each re-assignment. The best of all possible re-assignments is then executed and the algorithm iterates until there are no more good re-assignments or the maximum number of iterations is reached. In opttdev, the goodness-of-clustering statistic is total table deviance as calculated by `tabdev`. See the help file for `tabdev` for more detail.

## Value

a list which inherits from class 'opttdev', 'clustering' with components:

| | |
|---|---|
| `numitr` | the number of iterations performed |
| `dev` | a vector of total table deviance at each iteration of length 'numitr' |
| `clustering` | the vector of cluster memberships (as integers) for each sample |

## Note

Like many iterative re-assignment algorithms, opttdev is likely to be VERY slow from a random start or poor initial condition. opttdev is maybe better used to polish existing classifications

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## References

`http://ecology.msu.montana.edu/labdsv/R`

## See Also

`optpart`, `optindval`, `optsil`

## Examples

```
## Not run: data(shoshveg) # returns a data.frame of vegetation
## Not run: data(shoshsite)
## Not run: res <- opttdev(shoshveg,
                  as.numeric(cut(shoshsite$elevation,5)))
## End(Not run)
## Not run: # likely to be VERY slow
```

partana *Partition Analysis*

## Description

Partition analysis evaluates the within-cluster to among-cluster similarity of classifications as a measure of cluster validity

## Usage

```
partana(c,dist)
## S3 method for class 'partana'
summary(object, ...)
## S3 method for class 'partana'
plot(x,panel='all',zlim=range(x$ptc),col=heat.colors(12),...)
```

## Arguments

| | |
|---|---|
| `c` | an integer or factor vector, or an object of class 'clustering', 'partana', 'partition', or 'stride' |
| `dist` | an object of class 'dist' from functions `dist`, `dsvdis` or `vegdist`. |
| `object` | an object of class 'partana' |
| `x` | an object of class 'partana' |
| `panel` | an integer switch to indicate which panel to draw |
| `zlim` | the min and max values for the color map |
| `col` | a color map name (heat.colors(12) is the default) |
| `...` | ancillary arguments to pass to summary or plot |

## Details

Calculates mean object-to-cluster similarity, mean cluster-to-cluster similarity, and mean within-cluster to among-cluster similarity. partana operates on partitions or clusterings produced by a wide range of algorithms, including specific methods for the products of functions `optpart`, `slice`, `pam` and `diana`.

`summary` produces a matrix of the mean cluster-to-cluster similarities, and the overall within-cluster/among-cluster similarity ratio.

`plot` plots two panels in sequence in the current device. The first shows the mean similarity of every object to each cluster, sorted by mean similarity to the other members of its own cluster, with objects as columns and clusters as rows. The second panel shows the mean similarity of every cluster to every other cluster and mean within-cluster similarity, ignoring cluster size. These plots are known as 'Mondriaan' plots, where the similarities are given by lines colored from min to max. If the 'partana' object was produced by optpart, a third panel is plotted showing the trace of the optimization.

## Value

an object of class 'partana' with components:

| | |
|---|---|
| `ptc` | matrix of mean object-to-cluster similarity |
| `ctc` | matrix of mean cluster-to-cluster similarity |
| `clustering` | vector of numeric cluster assignments |
| `ratio` | within-cluster to among-cluster similarity ratio |

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## See Also

`partition`, `optpart`, `plot.partana`

## Examples

```
data(shoshveg)
data(shoshsite)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
demo.part <- partana(cut(shoshsite$elev,5),dis.bc)
summary(demo.part)
```

---

`partition` *Convert Object to Partition Object*

---

## Description

Convert an object of class 'partana' or class 'clustering' to an object of class 'partition'.

## Usage

```
partition(x, dist, ...)
```

## Arguments

| | |
|---|---|
| `x` | an object which inherits from class 'clustering' |
| `dist` | an object of class 'dist' |
| `...` | ancillary arguments to pass to 'partition' |

## Details

A 'partition' object is the output of several functions in package 'cluster'. This utility function converts objects from package 'optpart' to 'partitions' so that functions in that library are available.

### Value

an object of class ‘partition’ with components (and possibly others):

| | |
|---|---|
| `clustering` | vector of numeric cluster assignments |
| `silinfo` | a list with all silhouette information, only available when the number of clusters is non-trivial, i.e., $1 < k < n$. See ‘silhouette’ |

### Author(s)

David W. Roberts <`droberts@montana.edu`> `http://ecology.msu.montana.edu/labdsv/R`

### References

`http://ecology.msu.montana.edu/labdsv/R`

### See Also

`silhouette`, `partition`, `optpart`

### Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
part <- partana(sample(1:5,nrow(shoshveg),replace=TRUE),dis.bc)
result <- partition(part,dis.bc)
```

---

## phi — *Calculating the phi Statistic on Taxon Classifications*

---

### Description

Calculates the $phi$ statistic on a classified table of taxa

### Usage

```
phi(comm,clustering,minplt=10,p.adjust=FALSE)
```

### Arguments

| | |
|---|---|
| `comm` | a data.frame with samples as rows and attributes as columns |
| `clustering` | a vector of integers or an object of class ‘clustering’, ‘partition’, or ‘partana’ |
| `minplt` | the minimum number of samples a species must occur in to be included in the calculation |
| `p.adjust` | switch to control adjusting probabilities for simultaneous inference by Hochberg correction |

## Details

$phi$ is a statistic of agreement between two vectors. In this case the function calculates the distribution of each species within clusters of a partition, calculates the $phi$ statistic for each species in each cluster.

$$\phi = \frac{ad - bc}{\sqrt{(a+b) \times (c+d) \times (a+c) \times (b+c)}}$$

where:

| | |
|---|---|
| a | sample is in specified type and species is present |
| b | sample is not in group and species is present |
| c | sample is in type but species is not present |
| d | sample is not in type and species is not present |

## Value

A data.frame of $\phi$ values with species as rows and clusters as columns

## Author(s)

David W. Roberts <droberts@montana.edu>

## References

Tichy, L. and M. Chytry. 2006. Statistical determination of diagnostic species for site groups of unequal size. Journal of Vegetation Science 17:809-818.

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray')
opt.5 <- optpart(5,dis.bc)
phi(shoshveg,opt.5)
```

---

`refine` *Refining a Classification by Re-Assigning Memberships*

---

## Description

Refine allows you to re-assign specific elements of a classification from one class or cluster to another. In the default case, you simply interactively enter sample IDs and give a new cluster assignment. For PCO and NMDS ordinations, you do the assignments with a mouse.

## Usage

```
## Default S3 method:
refine(comm,clustering,...)
## S3 method for class 'dsvord'
refine(x,clustering,ax=1,ay=2,...)
```

## Arguments

| | |
|---|---|
| `comm` | a community data.frame |
| `x` | an ordination of class 'dsvord' |
| `clustering` | a clustering identity or membership vector |
| `ax` | the X axis of the ordination |
| `ay` | the Y axis of the ordination |
| `...` | ancillary arguments to allow differing numbers of arguments |

## Value

a list object of class 'clustering' with one component.

| | |
|---|---|
| `clustering` | a numeric vector giving the cluster assignment for each sample |

## Note

There are many, many ways to produce classifications in R, including several in package 'optpart'. refine is designed to take one of these classifications and polish it by making relatively few transfers. The ordination-based routines allow visual assessment of cluster validity, although in reduced dimensionality, which can be misleading.

## Author(s)

David W. Roberts <droberts@montana.edu>

## References

http://ecology.msu.montana.edu/labdsv/R

## Examples

```
## Not run: data(shoshveg)
## Not run: dis.bc <- dsvdis(shoshveg,'bray')
## Not run: opt.5 <- optpart(5,dis.bc)
## Not run: nmds.bc <- nmds(dis.bc)
## Not run: plot(nmds.bc)
## Not run: res <- refine(nmds.bc,opt.5)
```

## `reordclust` *Re-order Clusters in a Classification*

### Description

In it's simplest form simply reassigns cluster numbers in an existing classification to re-order tables and graphs. Can also be used to combine clusters into a fewer number of clusters.

### Usage

```
reordclust(clustering,from,to)
```

### Arguments

| | |
|---|---|
| `clustering` | a vector of (interger) cluster mmeberships, or an object of class 'clustering', 'partana', or 'partition' |
| `from` | an integer vector equal in length to the number of clusters that specifies the current clusters |
| `to` | an integer vector equal in length to the number of clusters that specifies the clusters the current clusters map to |

### Details

The function simply maps cluster numbers in the 'from' vector to the respective cluster number in the 'to' vector.

### Value

an object of class 'clustering'

### Note

As demonstrated in the examples below, reordclass can also combine existing clusters into fewer clusters while reordering if more than one cluster in the 'from' vector maps to the same cluster in the 'to' cluster.

### Author(s)

David W. Roberts <`droberts@montana.edu`>

### References

`http://ecology.msu.montana.edu/labdsv/R`

### See Also

`mergeclust`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray')
opt.10 <- optpart(10,dis.bc)
new <- reordclust(opt.10,1:10,c(1,3,5,7,9,2,4,6,8,10))
           # simply re-ordered
new2 <- reordclust(opt.10,1:10,c(1,1,2,2,3,3,4,4,5,5))
           # merge 1 and 2 into 1, 3 and 4 into 2, etc.
```

---

`shoshsite` *Site Data for the Shoshone National Forest, Wyoming, USA*

---

## Description

The coniferous forests of the Shoshone National Forest range from lower elevation woodlands dominated by *Pinus flexilis*, through forests of *Pseudotsuga menziesii, Pinus contorta, Picea engelmannii, Abies lasiocarpa* and *Pinus albicaulis* with increasing elevation (Steele et al. 1983). One hundred and fifty sample plots were chosen at random from a larger set for this data set; the larger set was stratified by elevation, exposure, surficial geology, and geographic distribution.

## Usage

```
data(shoshsite)
```

## Format

A data.frame with sample plots as rows, and site variable as columns. Variables comprise:

| | |
|---|---|
| elevation | elevation above sea level in meters |
| aspect | compass orientation of the site in degrees |
| slope | slope steepness in percent |
| av | aspect value: (cosd(aspect-30)+1)/2 |
| swb | site water balance: a tipping bucket model of water-year soil water |
| sprppt | spring precipitation in cm |
| sumppt | summer precipitation in cm |
| autppt | autumn precipitation in cm |
| winppt | winter precipitation in cm |
| sprtmp | spring mean temperature degrees C |
| sumtmp | summer mean temperature degrees C |
| auttmp | autumn mean temperature degrees C |
| winppt | winter mean temperature degrees C |
| sprpet | spring potential evapotranspiration in cm |
| sumpet | summer potential evapotranspiration in cm |
| autpet | autumn potential evapotranspiration in cm |
| winpet | winter potential evapotranspiration in cm |
| sprrad | spring direct and diffuse solar radiation (correcting for topographic shading) |
| sumrad | summer direct and diffuse solar radiation (correcting for topographic shading) |

| | |
|---|---|
| autrad | autumn direct and diffuse solar radiation (correcting for topographic shading) |
| win | winter direct and diffuse solar radiation (correcting for topographic shading) |
| ffd | frost free days |
| dday | degree days heat sum |
| tcol | mean monthly temperature of the coldest month |

## Note

The data were derived from a multi-year effort by numerous scientists and field technicians. The project was directed by Kent Houston, Soil Scientist and Ecologist, Shoshone National Forest. The site data were calculated primarily by Dr. Niklaus Zimmermann, WSL, Birmensdorf, Switzerland `http://www.wsl.ch/staff/niklaus.zimmermann/biophys.html`

## Source

Roberts, D.W. 2008. Statistical Analysis of Multidimensional Fuzzy Set Ordinations. Ecology 89:1246-1260

---

`shoshveg` *Vascular Plant Species Cover for the Shoshone National Forest, Wyoming, USA*

---

## Description

Percent cover (in codes) for 368 vascular plants on 150 $375m^2$ sample plots stratified across the Shoshone National Forest, Wyoming, USA. Plots were chosen at random from a larger dataset.

## Usage

```
data(shoshveg)
```

## Format

A data.frame with sample plots as rows and species as columns. Sample plots match the 'shoshsite' dataset.

The cover of all vascular plant species was recorded according to the following scale: present but $< 1\%$ = 0.1, 1-5% = 0.5, 5-15% = 1.0, 15-25% = 2.0, 25-35% = 3.0, 35-45% = 4.0, 45-55% = 5.0, 55-65% = 6.0, 65=75% = 7.0, 75-85% = 8.0. No species exhibited greater than 80% cover in the data set.

## Note

The data were derived from a multi-year effort by numerous scientists and field technicians. The project was directed by Kent Houston, Soil Scientist and Ecologist, Shoshone National Forest

## Source

Roberts, D.W. 2008. Statistical Analysis of Multidimensional Fuzzy Set Ordinations. Ecology 89:1246-1260.

---

`silhouette.partana` *Produce a Silhouette Object From a Partana, Clustering, or Stride Object*

---

## Description

Extracts components from a `partana`, `clustering`, or `stride` object, and passes the values to the `silhouette` function to produce an object of class silhouette.

## Usage

```
## S3 method for class 'partana'
silhouette(x, dist, ...)
## S3 method for class 'clustering'
silhouette(x, dist, ...)
## S3 method for class 'stride'
silhouette(x, dist, ...)
testsil(sil)
```

## Arguments

| | |
|---|---|
| `x` | an object of class ‘partana’, ‘clustering’, or ‘stride’ |
| `dist` | an object of class dist |
| `...` | miscellaneous arguments to pass to function silhouette |
| `sil` | an object of class ‘silhouette’ |

## Details

For ‘partana’ and ‘clustering’ objects the advantage over calling `silhouette` directly is that the row.names of the resulting object are added to the results, as opposed to consecutive integers.

For objects of class ‘stride’ the function extracts the component ‘clustering’ for each level of a stride object, and calls function `silhouette` in library ‘cluster’ returniung the mean silhouette width for each case.

`testsil` identifies ‘misfits’ in a partition, defined as plots with a negative silhouette width, and prints them out in a sorted list.

## Value

An object of class `silhouette`

## Note

This is a a simple conversion routine to allow plotting a silhouette plot for an object of class partana.

## Author(s)

David W. Roberts <droberts@montana.edu>

## References

http://ecology.msu.montana.edu/labdsv/R

## See Also

silhouette

## Examples

```
data(shoshveg)             # produces a data frame of vegetation data,
                           #samples as rows, attributes as columns
dis.bc <- dsvdis(shoshveg,'bray/curtis') # produces a Bray/Curtis
                                         # dissimilarity matrix
opt.5 <- optpart(5,dis.bc) # produces an optimal partitioning into
                           # 5 clusters
silhouette(opt.5,dis.bc)   # calculates the silhouette values
## Not run: plot(silhouette(opt.5,dis.bc)) # produce silhouette
                                         # plot on current device
```

---

`slice` *Slice a Hierarchical Clustering Dendrogram with a Mouse*

---

## Description

Allows a simple classification of objects by slicing a dendrogram of a hierarchical cluster analysis graphically with a mouse, or by simply giving a number.

## Usage

```
slice(clust, k=NULL)
```

## Arguments

| | |
|---|---|
| `clust` | an object of class ‘hclust’ produced by `hclust` |
| `k` | a desired number of clusters. If null, the function waits on a mouse click |

## Value

an object of class ‘clustering’, a list with a vector of cluster memberships

## Note

This function is a simple wrapper for `cutree` that allows users to click their mouse at the height they desire to slice the dendrogram, and to establish the result with a class of ‘clustering’ for ease of use in other functions. If you want to use the mouse, the dendrogram must have been previously plotted in the current graphic device.

## Author(s)

David W. Roberts <`droberts@montana.edu`> `http://ecology.msu.montana.edu/labdsv/R`

## See Also

`ordpart`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
demo <- hclust(dis.bc,'ave')
ave.5 <- slice(demo,5)
## Not run: plot(demo)
## Not run: ave.clust <- slice(demo)
```

---

`stride` *Stride: Producing a Sequence of Clusterings*

---

## Description

stride proceeds along a specified sequence creating clusterings or partitions of a dissimilarity matrix for each value of the sequence.

## Usage

```
stride(seq,arg2,type='pam',numrep=10,maxitr=100)
## S3 method for class 'stride'
plot(x, dist, col2=4, ...)
```

## Arguments

| | |
|---|---|
| `seq` | a sequence, in either a:b or seq(a,b,c) form |
| `arg2` | an object of class ‘dist’ from `dist`, `dsvdis` or `vegdist` among other sources, or of class ‘hclust’ |
| `type` | if arg2 is an object of class ‘dist’, type specifies the algorithm to produce clusters, and can be either ‘pam’ or ‘optpart’. |
| `numrep` | if arg2 is an object of class ‘dist’ and type = ‘optpart’, numrep specifies the number of replicates to run in function `optpart`. |

| | |
|---|---|
| `maxitr` | if arg2 is an object of class 'dist' and type = 'optpart' numrep specifies the maximum number of iterations per replicate in function `optpart` |
| `x` | an object of class 'stride' |
| `dist` | an object of class 'dist' from `dist`, `dsvdis`, or `vegdist` |
| `col2` | the color code for the second line in the graph |
| `...` | ancillary arguments to the plot function |

## Details

The specific action of function stride depends on the class of the second argument. If arg2 is of class 'dist', then clusters are generated by a fixed cluster algorithm. In this case, if type is 'pam', the function `pam` is called to produce the clusters. If type is 'optpart' the function `optpart` is called to produce the clusters. If arg2 is of class 'hclust', then the hlcust object is successively 'sliced' at levels specified by the sequence. The object of class 'hclust' can result from the function `hclust` using any of the methods provided, or by casting an object of class 'partition' to class 'hlcust' with the `as.hclust` function.

The default plot method for a stride plots the partana ratios (see `partana`) of each partition of the sequence on the left Y axis, and the silhouette widths of the same partitions on the right Y axis.

## Value

an object of class 'stride', which is a list with components:

| | |
|---|---|
| `clustering` | a data.frame with items as rows, and cluster IDs as columns, with one column for each value of the sequence |
| `seq` | a copy of the sequence employed |

## Author(s)

David W. Roberts `<droberts@montana.edu>`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray')
results <- stride(2:5,dis.bc)
```

## `tabdev` *Classification Validity Assessment by Table Deviance*

## Description

Table deviance is a method to assess the quality of classifications by calculating the clarity of the classification with respect to the original data, as opposed to a dissimilarity or distance matrix representation

## Usage

```
    ## Default S3 method:
tabdev(x,clustering,nitr=999,...)
    ## S3 method for class 'stride'
tabdev(x,taxa,...)
    ## S3 method for class 'tabdev'
summary(object,p=0.05,...)
```

## Arguments

| | |
|---|---|
| `x` | a matrix or data.frame of multivariate observations, with objects as rows, and attributes as columns |
| `clustering` | a vector of integer cluster assignments, or an object of class ‘clustering’ or ‘partana’ |
| `nitr` | number of iterations to perform in calculating the probability of obtaining as effective a classification as observed |
| `taxa` | a data.frame with samples as rows and species as columns |
| `object` | and object of class ‘tabdev’ |
| `p` | the maximum probability threshold to list species in the summary table |
| `...` | ancillary arguments to maintain compatibility with generic summary function |

## Details

Tabdev calculates the concentration of values within clusters. For each column, tabdev calculates the sum of values within classes and the sum within classes divided by the sum of that column to get fractional sums by class. These values are used to calculate the deviance of each row. Attributes that are widely dispersed among classes exhibit high deviance; attributes that are concentrated within a single class contribute zero deviance. An effective classification should exhibit low deviance.

Tabdev then permutes the values within columns and calculates the probability of observing as low a deviance as observed as $$ (m+1)/(niter + 1)$$ where $m$ is the number of cases with as low or lower deviance as observed.

## Value

a list with components:

| | |
|---|---|
| `spcdev` | a data.frame with species, deviance, and probability as columns |
| `totdev` | the total deviance of the entire table |

## Author(s)

David W. Roberts <`droberts@montana.edu`> `http://ecology.msu.montana.edu/labdsv/R`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
opt.5 <- optpart(5,dis.bc)
tabdev(shoshveg,opt.5)
```

---

## testpart *Identify Misclassified Plots in a Partition*

---

### Description

testopt analyzes the mean similarity of each sample to the cluster to which it is assigned to all other clusters, and lists those samples which have similarity higher to another cluster than to the one to which they are assigned.

### Usage

```
testpart(part,ord=TRUE)
```

### Arguments

| | |
|---|---|
| `part` | a object of class 'partana' from `partana` or `optpart` |
| `ord` | a switch to control whether the output is ordered |

### Details

Simply examines each sample plot, comparing the mean similarity of that sample to all other samples in the cluster to which it is assigned as compared to its mean similarity to all other clusters. Samples which are more similar to other clusters than to the one to which they are assigned are listed in a table which gives their current cluster assignment, the cluster to which they are more similar, and the mean similarities of that sample to all clusters.

If 'ord=TRUE' then the output is ordered to reflect target clusters.

### Value

a table of values

### Author(s)

David W. Roberts <`droberts@montana.edu`>

### See Also

`partana`, `partana`, `silhouette`

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray/curtis')
opt.5 <- optpart(5,dis.bc)
testpart(opt.5)
```

---

`typal` *Identification of Typal Samples in a Partition*

---

## Description

Identifies samples that typify clusters in a partition based on dissimilarity.

## Usage

```
typal(clustering,dist,k=1)
```

## Arguments

| | |
|---|---|
| `clustering` | a vector of integers or an object of class ‘clustering’, ‘partition’, or ‘partana’ |
| `dist` | and object of class ‘dist’ from `dist`, `dsvdis`, or `vegdist` |
| `k` | number of typal species/cluster to identify |

## Details

typal calculates two versions of typal species based on silhouette analysis (see `silhouette`) and partana analysis (see `partana`). With respect to silhouette analysis, the function returns k species with the largest positive silhouette width for each cluster. With respect to the partana analysis the function returns the k species with the highest mean similarity to the cluster.

## Value

Returns a list with two data.frames. The first, partana, gives the clusters as rows and typal samples as columns from the perspective of the partana ratio. The second, silhouette, also gives the clusters as rows and typal samples as columns but from the perspective of silhouette widths.

## Author(s)

David W. Roberts <`droberts@montana.edu`>

## Examples

```
data(shoshveg)
dis.bc <- dsvdis(shoshveg,'bray')
opt.5 <- bestopt(dis.bc,5,20)
typal(opt.5,dis.bc,3)
```

# Index