Package 'TraMineR'

November 25, 2015

Version 1.8-11

Date 2015-11-25

Title Trajectory Miner: a Toolbox for Exploring and Rendering
Sequences

Depends R (>= 2.8.1),

Imports utils, RColorBrewer, boot, graphics, grDevices, stats, Hmisc

Suggests cluster, xtable

Description Toolbox for the manipulation, description and rendering of sequences, and more generally the mining of sequence data in the field of social sciences. Although the toolbox is primarily intended for analyzing state or event sequences that describe life courses such as family formation histories or professional careers, its features also apply to many other kinds of categorical sequence data. It accepts many different sequence representations as input and provides tools for converting sequences from one format to another. It offers several functions for describing and rendering sequences, for computing distances between sequences with different metrics (among which optimal matching), original dissimilarity-based analysis tools, and simple functions for extracting the most frequent subsequences and identifying the most discriminating ones among them. A user's guide can be found on the TraMineR web page.

License GPL (>= 2)

URL http://mephisto.unige.ch/traminer

Encoding latin1

Maintainer Gilbert Ritschard <gilbert.ritschard@unige.ch>

NeedsCompilation yes

Author Alexis Gabadinho [aut, cph],
Matthias Studer [aut, cph],
Nicolas Muller [aut],
Reto Buergin [aut],
Gilbert Ritschard [aut, cre, cph]

Repository CRAN

Date/Publication 2015-11-25 13:49:07

R **topics documented:**

TraMineR-package   *Trajectory Miner: a Toolbox for Exploring and Rendering Sequences*

Description

(Version: 1.8.11) Toolbox for the manipulation, description and rendering of sequences, and more generally the mining of sequence data in the field of social sciences. Although the toolbox is primarily intended for analyzing state or event sequences that describe life courses such as family formation histories or professional careers, its features also apply to many other kinds of categorical sequence data. It accepts many different sequence representations as input and provides tools for converting sequences from one format to another. It offers several functions for describing and rendering sequences, for computing distances between sequences with different metrics (among which optimal matching), original dissimilarity-based analysis tools, and simple functions for extracting the most frequent subsequences and identifying the most discriminating ones among them. A user's guide can be found on the TraMineR web page.

Details

TraMineR provides tools for both state sequences and event sequences. The first step when using the package is to define a state sequence object (with `seqdef`) if you want to explore state sequences, and an event sequence object (with `seqecreate`) if you are interested in event sequencing.

State sequences are defined from a series of variables giving the states at the successive positions, while event sequences are defined from (vertical) time stamped event data. The package, however, can handle many other different data organizations and provides tools to help converting state sequences into event sequences and vice versa.

Author(s)

Alexis Gabadinho, Matthias Studer, Nicolas S. Muller, Reto Buergin, and Gilbert Ritschard

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2009). Mining Sequence Data in R with the `TraMineR` package: A user's guide. Department of Econometrics and Laboratory of Demography, University of Geneva

Examples

```
## load the mvad data
library(TraMineR)
data(mvad)

## create a state sequence object from columns 17 to 86
mvad.seq <- seqdef(mvad[,17:86])
```

```
## distribution plot by sex (male)
seqdplot(mvad.seq, group=mvad$male, border=NA)

## compute the LCS pairwise distance matrix
## among the first 10 sequences
mvad.lcs <- seqdist(mvad.seq[1:10,], method="LCS")
```

---

actcal   *Example data set: Activity calendar from the Swiss Household Panel*

Description

This data set contains 2000 individual sequences of monthly activity statuses from January to December 2000.

Usage

```
data(actcal)
```

Format

A data frame with 2000 rows, 12 state variables, 1 id variable and 11 covariates.

Details

The data set is a subsample of the data collected by the Swiss Household Panel (SHP).

The state column (variable) names are 'jan00', 'feb00', etc... and correspond to columns 13 to 24.

There are four possible states:

A = Full-time paid job (> 37 hours)
B = Long part-time paid job (19-36 hours)
C = Short part-time paid job (1-18 hours)
D = Unemployed (no work)

The data set contains also the following covariates:

age00	(age in 2000)
educat00	(education level)
civsta00	(civil status)
nbadul00	(number of adults in household)
nbkid00	(number of children)
aoldki00	(age of oldest kid)
ayouki00	(age of youngest kid)
region00	(residence region)
com2.00	(residence commune type)
sex	(sex of respondent)
birthy	(birth year)

Source

Swiss Household Panel

References

www.swisspanel.ch

---

actcal.tse	*Example data set: Activity calendar from the Swiss Household Panel (time stamped event format)*

Description

This data set contains events defined from the state sequences in the actcal data set. It was created with the code shown in the examples section. It is provided to symplify example of event sequence mining.

Usage

```
data(actcal.tse)
```

Format

Time stamped events derived from state sequences in the actcal data set.

Source

Swiss Household Panel

See Also

seqformat, actcal

Examples

```
data(actcal)
actcal.seq <- seqdef(actcal[,13:24])

## Defining the transition matrix
transition <- seqetm(actcal.seq, method="transition")
transition[1,1:4] <- c("FullTime"          , "Decrease,PartTime",
      "Decrease,LowPartTime", "Stop")
transition[2,1:4] <- c("Increase,FullTime", "PartTime"          ,
      "Decrease,LowPartTime", "Stop")
transition[3,1:4] <- c("Increase,FullTime", "Increase,PartTime",
     "LowPartTime"         , "Stop")
transition[4,1:4] <- c("Start,FullTime"   , "Start,PartTime"    ,
     "Start,LowPartTime"    , "NoActivity")
transition
```

```
## Converting STS data to TSE
actcal.tse <- seqformat(actcal,var=13:24, from='STS',to='TSE',
tevent=transition)

## Defining the event sequence object
actcal.seqe <- seqecreate(id=actcal.tse$id,
time=actcal.tse$time, event=actcal.tse$event)
```

---

alphabet   *Get or set the alphabet of a sequence object*

Description

This function gets or sets the (short) labels associated to the states in the alphabet of a sequence object (the list of all possible states, some of which states may not appear in the data).

Usage

```
alphabet(seqdata)
alphabet(seqdata) <- value
```

Arguments

seqdata   a state sequence object as defined with the seqdef function.

value   a character vector of the same length as the vector returned by the alphabet function, i.e. one label for each state in the alphabet.

Details

A state sequence object — created with the seqdef function — stores sequences as a matrix where columns are factors. The levels of the factors are made of the alphabet as well as the codes for missing value and void elements. The alphabet function retrieves or sets the "alphabet" attribute of the sequence object. The state names composing the alphabet are preferably short labels, since they are used for printing sequences. Longer labels for describing more precisely each state in legend are stored in the "labels" attribute of the sequence object.

Value

For 'alphabet' a character vector containing the alphabet.

For 'alphabet <-' the updated sequence object.

Author(s)

Alexis Gabadinho

See Also

seqdef

Examples

```
## Creating a sequence object with the columns 13 to 24
## in the 'actcal' example data set
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## Retrieving the alphabet
alphabet(actcal.seq)

## Setting the alphabet
alphabet(actcal.seq) <- c("FT", "PT", "LT", "NO")
```

---

biofam	*Example data set: Family life states from the Swiss Household Panel biographical survey*

Description

2000 16 year-long family life sequences built from the retrospective biographical survey carried out by the Swiss Household Panel (SHP) in 2002.

Usage

```
data(biofam)
```

Format

A data frame with 2000 rows, 16 state variables, 1 id variable and 7 covariates and 2 weights variables.

Details

The *biofam* data set was constructed by Müller et al. (2007) from the data of the retrospective biographical survey carried out by the Swiss Household Panel (SHP) in 2002.

The data set contains (in columns 10 to 25) sequences of family life states from age 15 to 30 (sequence length is 16) and a series of covariates. The sequences are a sample of 2000 sequences of those created from the SHP biographical survey. It includes only individuals who were at least 30 years old at the time of the survey. The *biofam* data set describes family life courses of 2000 individuals born between 1909 and 1972.

The states numbered from 0 to 7 are defined from the combination of five basic states, namely Living with parents (Parent), Left home (Left), Married (Marr), Having Children (Child), Divorced:

0 = "Parent"
1 = "Left"
2 = "Married"
3 = "Left+Marr"
4 = "Child"

5 = "Left+Child"
6 = "Left+Marr+Child"
7 = "Divorced"

The covariates are:

```
sex
birthyr    (birth year)
nat_1_02   (first nationality)
plingu02   (language of questionnaire)
p02r01     (religion)
p02r04     (religious participation)
cspfaj     (father's social status)
cspmoj     (mother's social status)
```

Two additional weights variables are inserted for illustrative purpose ONLY (since biofam is a sub-sample of the original data, these weights are not adapted to the actual data):

```
wp00tbgp   (weights inflating to the Swiss population)
wp00tbgs   (weights respecting sample size)
```

Source

Swiss Household Panel <www.swisspanel.ch>

References

Müller, N. S., M. Studer, G. Ritschard (2007). Classification de parcours de vie à l'aide de l'optimal matching. In *XIVe Rencontre de la Société francophone de classification (SFC 2007), Paris, 5 - 7 septembre 2007*, pp. 157–160.

---

cpal   *Get or set the color palette of a sequence object*

Description

This function gets or sets the color palette of a sequence object, that is, the list of colors used to represent the states.

Usage

```
cpal(seqdata)
cpal(seqdata) <- value
```

Arguments

seqdata   a state sequence object as defined by the [seqdef](#) function.

value   a vector containing the colors, of length equal to the number of states in the al-
               phabet. The colors can be passed as character strings representing color names
               such as returned by the [colors](#) function, as hexadecimal values or as RGB vec-
               tors using the [rgb](#) function. Each color is attributed to the corresponding state
               in the alphabet, the order being the one returned by the [alphabet](#).

Details

In the plot functions provided for visualizing sequence objects, a different color is associated to
each state of the alphabet. The color palette is defined when creating the sequence object, either
automatically using the brewer.pal function of the RColorBrewer package or by specifying a user
defined color vector. The cpal function can be used to get or set the color palette of a previously
defined sequence object.

Value

For 'cpal(seqdata)' a vector containing the colors.

For 'cpal(seqdata) <-' the updated sequence object.

Author(s)

Alexis Gabadinho

See Also

[seqdef](#)

Examples

```
## Creating a sequence object with the columns 13 to 24
## in the 'actcal' example data set
## The color palette is automatically set
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## Retrieving the color palette
cpal(actcal.seq)
seqiplot(actcal.seq)

## Setting a user defined color palette
cpal(actcal.seq) <- c("blue","red", "green", "yellow")
seqiplot(actcal.seq)
```

---

dissassoc   *Analysis of discrepancy from dissimilarity measures*

Description

Compute and test the share of discrepancy (defined from a dissimilarity matrix) explained by a categorical variable.

Usage

```
dissassoc(diss, group, weights=NULL, R=1000,
          weight.permutation="replicate", squared=FALSE)
```

Arguments

diss	A dissimilarity matrix or a dist object (see [dist](#))
group	A categorical variable. For a numerical variable use [dissmfac](#).
weights	optional numerical vector containing weights.
R	Number of permutations for computing the p-value. If equal to 1, no permutation test is performed.
weight.permutation	
	Weighted permutation method: `"diss"` (attach weights to the dissimilarity matrix), `"replicate"` (replicate case using `weights`), `"rounded-replicate"` (replicate case using rounded `weights`), `"random-sampling"` (random assignment of covariate profiles to the objects using distributions defined by the weights.)
squared	Logical. If `TRUE` the dissimilarities `diss` are squared.

Details

The `dissassoc` function assesses the association between objects characterized by their dissimilarity matrix and a discrete covariate. It provides a generalization of the ANOVA principle to any kind of distance metric. The function returns a pseudo R-square that can be interpreted as a usual R-square. The statistical significance of the association is computed by means of permutation tests. The function performs also a test of discrepancy homogeneity (equality of within variances) using a generalization of the Levene statistic and Bartlett's statistics.

There are `print` and `hist` methods (the latter producing an histogram of the permuted values used for testing the significance).

If a numeric `group` variable is provided, it will be treated as categorical, i.e., each different value will be considered as a different category. To measure the 'linear' effect of a numerical variable, use [dissmfac](#).

Value

An object of class `dissassoc` with the following components:

groups	A data frame with the number of cases and the discrepancy of each group

anova.table	The pseudo ANOVA table
stat	The value of the statistics and their p-values
perms	The permutation object, containing the values computed for each permutation

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

References

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2011). Discrepancy analysis of state sequences, *Sociological Methods and Research*, Vol. 40(3), 471-510.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2010) Discrepancy analysis of complex objects using dissimilarities. In F. Guillet, G. Ritschard, H. Briand, and D. A. Zighed (Eds.), *Advances in Knowledge Discovery and Management*, Studies in Computational Intelligence, Volume 292, pp. 3-19. Berlin: Springer.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2009). Analyse de dissimilarités par arbre d'induction. In EGC 2009, *Revue des Nouvelles Technologies de l'Information*, Vol. E-15, pp. 7–18.

Anderson, M. J. (2001) A new method for non-parametric multivariate analysis of variance. *Austral Ecology* **26**, 32–46.

Batagelj, V. (1988) Generalized Ward and related clustering problems. In H. Bock (Ed.), *Classification and related methods of data analysis*, Amsterdam: North-Holland, pp. 67–74.

See Also

[dissvar](#) to compute the pseudo variance from dissimilarities and for a basic introduction to concepts of pseudo variance analysis.
[disstree](#) for an induction tree analyse of objects characterized by a dissimilarity matrix.
[disscenter](#) to compute the distance of each object to its group center from pairwise dissimilarities.
[dissmfac](#) to perform multi-factor analysis of variance from pairwise dissimilarities.

Examples

```
## Defining a state sequence object
data(mvad)
mvad.seq <- seqdef(mvad[, 17:86])

## Building dissimilarities (any dissimilarity measure can be used)
mvad.ham <- seqdist(mvad.seq, method="HAM")

## R=1 implies no permutation test
da <- dissassoc(mvad.ham, group=mvad$gcse5eq, R=10)
print(da)
hist(da)
```

---

disscenter *Compute distances to the center of a group*

Description

Computes the dissimilarity between objects and their group center from their pairwise dissimilarity matrix.

Usage

```
disscenter(diss, group=NULL, medoids.index=NULL,
          allcenter = FALSE, weights=NULL, squared=FALSE)
```

Arguments

diss   a dissimilarity matrix such as generated by [seqdist](#), or a dist object (see [dist](#))

group   if NULL (default), the whole data set is considered. Otherwise a different center is considered for each distinct value of the group variable

medoids.index if NULL, returns the dissimilarity to the center. If set to "first", returns the index of the first encountered most central sequence. If group is set, an index is returned per group. When set to "all", indexes of all medoids (one list per group) are returned.

allcenter   logical. If TRUE, returns a data.frame containing the dissimilarity between each object and its group center, each column corresponding to a group.

weights   optional numerical vector containing weights.

squared   Logical. If TRUE diss is squared.

Details

This function computes the dissimilarity between given objects and their group center. It is possible that the group center does not belong to the space formed by the objects (in the same way as the average of integer numbers is not necessarily an integer itself). This distance can also be understood as the contribution to the discrepancy (see [dissvar](#)). Note that when the dissimilarity measure does not respect the triangle inequality, the dissimilarity between a given object and its group center may be negative

It can be shown that this dissimilarity is equal to (see *Batagelj 1988*):

$$d_{x\tilde{g}} = \frac{1}{n} \Big(\sum_{i=1}^{n} d_{xi} - SS \Big)$$

where SS is the sum of squares (see [dissvar](#)).

Value

A vector with the dissimilarity to the group center for each object, or a list of medoid indexes.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

References

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2011). Discrepancy analysis of state sequences, *Sociological Methods and Research*, Vol. 40(3), 471-510.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2010) Discrepancy analysis of complex objects using dissimilarities. In F. Guillet, G. Ritschard, D. A. Zighed and H. Briand (Eds.), *Advances in Knowledge Discovery and Management*, Studies in Computational Intelligence, Volume 292, pp. 3-19. Berlin: Springer.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2009) Analyse de dissimilarités par arbre d'induction. In EGC 2009, *Revue des Nouvelles Technologies de l'Information*, Vol. E-15, pp. 7–18.

Batagelj, V. (1988) Generalized ward and related clustering problems. In H. Bock (Ed.), *Classification and related methods of data analysis*, Amsterdam: North-Holland, pp. 67–74.

See Also

[dissvar](#) to compute the pseudo variance from dissimilarities and for a basic introduction to concepts of pseudo variance analysis
[dissassoc](#) to test association between objects represented by their dissimilarities and a covariate.
[disstree](#) for an induction tree analyse of objects characterized by a dissimilarity matrix.
[dissmfac](#) to perform multi-factor analysis of variance from pairwise dissimilarities.

Examples

```
## Defining a state sequence object
data(mvad)
mvad.seq <- seqdef(mvad[, 17:86])

## Building dissimilarities (any dissimilarity measure can be used)
mvad.ham <- seqdist(mvad.seq, method="HAM")

## Compute distance to center according to group gcse5eq
dc <- disscenter(mvad.ham, group=mvad$gcse5eq)

## Ploting distribution of dissimilarity  to center
boxplot(dc~mvad$gcse5eq, col="cyan")

## Retrieving index of the first medoids, one per group
dc <- disscenter(mvad.ham, group=mvad$Grammar, medoids.index="first")
print(dc)

## Retrieving index of all medoids in each group
dc <- disscenter(mvad.ham, group=mvad$Grammar, medoids.index="all")
print(dc)
```

---

dissmfac   *Multi-factor ANOVA from a dissimilarity matrix*

Description

Perform a multi-factor analysis of variance from a dissimilarity matrix.

Usage

```
dissmfacw(formula, data, R = 1000, gower = FALSE, squared = FALSE,
    weights = NULL)
dissmfac(formula, data, R = 1000, gower = FALSE, squared = TRUE,
    permutation = "dissmatrix")
```

Arguments

formula	A regression-like formula. The left hand side term should be a dissimilarity matrix or a `dist` object.
data	A data frame from which the variables in `formula` should be taken.
R	Number of permutations used to assess significance.
gower	Logical: Is the dissimilarity matrix already a Gower matrix?
squared	Logical: Should we square the provided dissimilarities?
weights	Optional numerical vector of case weights.
permutation	Deprecated. Kept for backward compatibility.

Details

This method is, in some way, a generalization of [dissassoc](#) to account for several explanatory variables. The function computes the part of discrepancy explained by the list of covariates specified in the `formula`. It provides for each covariate the Type-II effect, i.e. the effect measured when removing the covariate from the full model with all variables included.

(The returned F values may slightly differ from those obtained with TraMineR versions older than 1.8-9. Since 1.8-9, the within sum of squares at the denominator is divided by $n - m$ instead of $n - m - 1$, where n is the sample size and m the total number of predictors and/or contrasts used to represent categorical factors.)

For a single factor dissmfac is slower than [dissassoc](#). Moreover, the latter performs also tests for homogeneity in within-group discrepancies (equality of variances) with a generalization of Levene's and Bartlett's statistics.

Part of the function is based on the Multivariate Matrix Regression with qr decomposition algorithm written in SciPy-Python by Ondrej Libiger and Matt Zapala (See *Zapala and Schork*, 2006, for a full reference.) The algorithm has been adapted for Type-II effects and extended to account for case weights.

Value

A `dissmultifactor` object with the following components:

mfac	The part of variance explained by each variable (comparing full model to model without the specified variable) and its significance using permutation test
call	Function call
perms	Permutation values as a `boot` object

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

References

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2011). Discrepancy analysis of state sequences, *Sociological Methods and Research*, Vol. 40(3), 471-510.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2010) Discrepancy analysis of complex objects using dissimilarities. In F. Guillet, G. Ritschard, D. A. Zighed and H. Briand (Eds.), *Advances in Knowledge Discovery and Management*, Studies in Computational Intelligence, Volume 292, pp. 3-19. Berlin: Springer.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2009). Analyse de dissimilarités par arbre d'induction. In EGC 2009, *Revue des Nouvelles Technologies de l'Information*, Vol. E-15, pp. 7-18.

Anderson, M. J. (2001). A new method for non-parametric multivariate analysis of variance. *Austral Ecology* 26, 32-46.

McArdle, B. H. and M. J. Anderson (2001). Fitting multivariate models to community data: A comment on distance-based redundancy analysis. *Ecology* 82(1), 290-297.

Zapala, M. A. and N. J. Schork (2006). Multivariate regression analysis of distance matrices for testing associations between gene expression patterns and related variables. *Proceedings of the National Academy of Sciences of the United States of America* 103(51), 19430-19435.

See Also

[dissvar](#) to compute a pseudo variance from dissimilarities and for a basic introduction to concepts of discrepancy analysis.
[dissassoc](#) to test association between objects represented by their dissimilarities and a covariate.
[disstree](#) for an induction tree analysis of objects characterized by a dissimilarity matrix.
[disscenter](#) to compute the distance of each object to its group center from pairwise dissimilarities.

Examples

```
## Define the state sequence object
data(mvad)
mvad.seq <- seqdef(mvad[, 17:86])

## Compute dissimilarities (any dissimilarity measure can be used)
mvad.ham <- seqdist(mvad.seq, method="HAM")
```

```
## And now the multi-factor analysis
print(dissmfac(mvad.ham ~ male + Grammar + funemp +
gcse5eq + fmpr + livboth, data=mvad, R=10))
```

---

dissrep   *Extracting sets of representative objects using a dissimilarity matrix*

Description

The function extracts a set of representative objects that exhibits the key features of the whole data set, the goal being to get easy sounded interpretation of the latter. The user can set either the desired coverage level (the proportion of objects having a representative in their neighborhood) or the desired number of representatives.

Usage

```
dissrep(diss, criterion="density",
    score=NULL, decreasing=TRUE,
    trep=0.25, nrep=NULL, tsim=0.1, dmax=NULL, weights=NULL)
```

Arguments

diss	A dissimilarity matrix or a dist object (see [dist](#))
criterion	the representativeness criterion for sorting the candidate list. One of "freq" (frequency), "density" (neighborhood density) or "dist" (centrality). An optional vector containing the scores for sorting the candidate objects may also be provided. See below and details.
score	an optional vector containing the representativeness scores used for sorting the objects in the candidate list. The length of the vector must be equal to the number of rows/columns in the distance matrix, i.e the number of objects.
decreasing	if a score vector is provided, indicates whether the objects in the candidate list must be sorted in ascending or decreasing order of this score. The first object in the candidate list is supposed to be the most representative.
trep	controls the size of the representative set by setting the desired coverage level, i.e the proportion of objects having a representative in their neighborhood. Neighborhood radius is defined by tsim.
nrep	number of representatives. If NULL (default), trep argument is used to control the size of the representative set.
tsim	neighborhood radius as a percentage of the maximum (theoretical) distance dmax. Defaults to 0.1 (10%). Object y is redundant to object x when it is in the neighborhood of x, i.e., within a distance tsim*dmax from x.
dmax	maximum theoretical distance. Used to derive the neighborhood radius as tsim*dmax. If NULL, the value of dmax is derived from the dissimilarity matrix.
weights	vector of weights of length equal to the number of rows of the dissimilarity matrix. If NULL, equal weights are assigned.

Details

The representative set is obtained by an heuristic. Representatives are selected by successively extracting from the sequences sorted by their representativeness score those which are not redundant with already retained representatives. The selection stops when either the desired coverage or the wanted number of representatives is reached. Objects are sorted either by the values provided as score argument, or by specifying one of the following as criterion argument: "freq" (*sequence frequency*), "density" (*neighborhood density*), "dist" (*centrality*).

The *frequency* criterion uses the frequencies as representativeness score. The frequency of an object in the data is computed as the number of other objects with whom the dissimilarity is equal to 0. The more frequent an object the more representative it is supposed to be. Hence, objects are sorted in decreasing frequency order. Indeed, this criterion is the neighborhood (see below) criterion with the neighborhood diameter set to 0.

The *neighborhood density* is the number—density—of sequences in the neighborhood of the object. This requires to set the neighborhood radius tsim. Objects are sorted in decreasing density order.

The *centrality* criterion is the sum of distances to all other objects. The smallest the sum, the most representative the sequence.

Use criterion="dist" and nrep=1 to get the medoid and criterion="density" and nrep=1 to get the densest object pattern.

For more details, see *Gabadinho et al., 2011.*

Value

An object of class diss.rep. This is a vector containing the indexes of the representative objects with the following additional attributes:

Scores	a vector with the representative score of each object given the chosen criterion.
Distances	a matrix with the distance of each object to its nearest representative.
Statistics	a data frame with quality measures for each representative: number of objects attributed to the representative, number of object in the representative's neighborhood, mean distance to the representative.
Quality	overall quality measure.

Print and summary methods are available.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

References

Gabadinho A, Ritschard G (2013). "Searching for typical life trajectories applied to child birth histories", In R Lévy, E. Widmer (eds.), *Gendered Life Courses*, pp. 287-312. Vienna: LIT.

Gabadinho A, Ritschard G, Studer M, Müller NS (2011). "Extracting and Rendering Representative Sequences", In A Fred, JLG Dietz, K Liu, J Filipe (eds.), *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 128 of *Communications in Computer and Information Science (CCIS)*, pp. 94-106. Springer-Verlag.

See Also

seqrep, disscenter

Examples

```
## Defining a sequence object with the data in columns 10 to 25
## (family status from age 15 to 30) in the biofam data set
data(biofam)
biofam.lab <- c("Parent", "Left", "Married", "Left+Marr",
"Child", "Left+Child", "Left+Marr+Child", "Divorced")
biofam.seq <- seqdef(biofam, 10:25, labels=biofam.lab)

## Computing the distance matrix
costs <- seqsubm(biofam.seq, method="TRATE")
biofam.om <- seqdist(biofam.seq, method="OM", sm=costs)

## Representative set using the neighborhood density criterion
biofam.rep <- dissrep(biofam.om)
biofam.rep
summary(biofam.rep)
```

---

disstree   *Dissimilarity Tree*

Description

Tree structured discrepancy analysis of objects described by their pairwise dissimilarities.

Usage

```
disstree(formula, data = NULL, weights = NULL, minSize = 0.05,
         maxdepth = 5, R = 1000, pval = 0.01, object = NULL,
         weight.permutation = "replicate", squared = FALSE,
         first = NULL)
```

Arguments

formula	Formula with a dissimilarity matrix as left hand side and the candidate partitioning variables on the right side.
data	Data frame where variables in formula will be searched for.
weights	Optional numerical vector of weights.
minSize	Minimum number of cases in a node, will be treated as a proportion if less than 1.
maxdepth	Maximum depth of the tree
R	Number of permutations used to assess the significance of the split.
pval	Maximum allowed p-value for a split

object	An optional R object represented by the dissimilarity matrix. This object may be used by the `print` method or `disstree2dot` to render specific object type.
weight.permutation	
	Weight permutation method: `"diss"` (attach weights to the dissimilarity matrix), `"replicate"` (replicate cases using `weights`), `"rounded-replicate"` (replicate case using rounded `weights`), `"random-sampling"` (random assignment of covariate profiles to the objects using distributions defined by the weights.)
squared	Logical: Should the `diss` dissimilarities be squared?
first	One of the variable in the right-hand side of the formula. This forces the first node of the tree to be split by this variable.

Details

The procedure iteratively splits the data. At each step, the procedure selects the variable and split that explain the greatest part of the discrepancy, i.e., the split for which we get the highest pseudo R2. The significance of the retained split is assessed through a permutation test.

`seqtree` provides a simpler interface if you plan to use `disstree` for state sequence objects.

Value

An object of class `disstree` that contains the following components:

root	A node object, root of the tree
info	General information such as parameters used to build the tree
info$adjustment	
	A `dissassoc` object providing global statistics for tree.
formula	The formula used to generate the tree
data	data used to build the tree
weights	weights

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

References

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2011). Discrepancy analysis of state sequences, *Sociological Methods and Research*, Vol. 40(3), 471-510.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2010) Discrepancy analysis of complex objects using dissimilarities. In F. Guillet, G. Ritschard, D. A. Zighed and H. Briand (Eds.), *Advances in Knowledge Discovery and Management*, Studies in Computational Intelligence, Volume 292, pp. 3-19. Berlin: Springer.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2009) Analyse de dissimilarités par arbre d'induction. In EGC 2009, *Revue des Nouvelles Technologies de l'Information*, Vol. E-15, pp. 7-18.

Anderson, M. J. (2001) A new method for non-parametric multivariate analysis of variance. *Austral Ecology* **26**, 32-46.

Batagelj, V. (1988) Generalized ward and related clustering problems. In H. Bock (Ed.), *Classification and related methods of data analysis*, Amsterdam: North-Holland, pp. 67-74.

Piccarreta, R. et F. C. Billari (2007) Clustering work and family trajectories by using a divisive algorithm. *Journal of the Royal Statistical Society A* **170**(4), 1061–1078.

See Also

seqtree to generate a specific disstree objects for analyzing state sequences.
seqtreedisplay to generate graphic representation of seqtree objects when analyzing state sequences.
disstreedisplay is a more general interface to generate such representation for other type of objects.
dissvar to compute discrepancy using dissimilarities and for a basic introduction to discrepancy analysis.
dissassoc to test association between objects represented by their dissimilarities and a covariate.
dissmfac to perform multi-factor analysis of variance from pairwise dissimilarities.
disscenter to compute the distance of each object to its group center from pairwise dissimilarities.

Examples

```
data(mvad)

## Defining a state sequence object
mvad.seq <- seqdef(mvad[, 17:86])

## Computing dissimilarities (any dissimilarity measure can be used)
mvad.ham <- seqdist(mvad.seq, method="HAM")
dt <- disstree(mvad.ham~ male + Grammar + funemp + gcse5eq + fmpr + livboth,
    data=mvad, R = 10)
print(dt)


## Will only work if GraphViz is properly installed
## See seqtree for simpler way to plot a sequence tree.
## Not run:
disstreedisplay(dt, imagefunc=seqdplot, imagedata=mvad.seq,
## Additional parameters passed to seqdplot
withlegend=FALSE, axes=FALSE, ylab="")

## End(Not run)
## Second method, using a specific function
myplotfunction <- function(individuals, seqs, ...) {
par(font.sub=2, mar=c(3,0,6,0), mgp=c(0,0,0))
## using mds to order sequence in seqiplot
mds <- cmdscale(seqdist(seqs[individuals,], method="HAM"),k=1)
seqiplot(seqs[individuals,], sortv=mds,...)
}

## If imagedata is not set, index of individuals are sent to imagefunc
## Not run:
disstreedisplay(dt, imagefunc=myplotfunction, title.cex=3,
```

```
## additional parameters passed to myplotfunction
seqs=mvad.seq,
## additional parameters passed to seqiplot (through myplotfunction)
withlegend=FALSE, axes=FALSE, tlim=0, space=0, ylab="", border=NA)

## End(Not run)
```

---

disstree2dot   *Graphical representation of a dissimilarity tree*

Description

Functions to generate a "dot" file and associated images files that can be used in GraphViz to get a
graphical representation of the tree.

Usage

```
disstree2dot(tree, filename, digits=3, imagefunc=NULL, imagedata=NULL,
             imgLeafOnly=FALSE, devicefunc="jpeg", imageext="jpg",
             device.arg=list(), use.title=TRUE, label.loc="main",
             node.loc="main", split.loc="sub", title.cex=1, legendtext=NULL,
             legendimage=NULL, qualityimage=NULL, showdepth=FALSE,
             title.outer=FALSE, ...)

disstree2dotp(tree, filename, imagedata=NULL, imgLeafOnly=FALSE,
             imagefunc=plot, title.cex=3, withquality=TRUE,
             quality.fontsize=title.cex, title.outer=FALSE, ...)

seqtree2dot(tree, filename, seqdata=tree$info$object, imgLeafOnly=FALSE,
             sortv=NULL, dist.matrix=NULL, title.cex=3, withlegend="auto",
             legend.fontsize=title.cex, withquality=FALSE,
             quality.fontsize=title.cex, axes=FALSE, ...)
```

Arguments

tree	The tree to be plotted.
filename	A filename, without extension, that will be used to generate image and dot files.
digits	Number of significant digits to plot.
imagefunc	A function to plot the individuals in a node, see details.
imagedata	a data.frame that will be passed to imagefunc, see details.
imgLeafOnly	Logical: If TRUE, only terminal node will be plotted.
devicefunc	A device function, "jpeg" by default.
imageext	extension for image files.
device.arg	Argument passed to devicefunc.

use.title	Logical: If TRUE, node information will be printed using title command, see details.
label.loc	Location of the node label, see title for possible values.
node.loc	Node content location, see title for possible values.
split.loc	Split information location, see title for possible values.
title.cex	cex applied to all calls to title (see use.title).
title.outer	Logical: If TRUE, the title (see use.title) is printed in the outer margins.
legendtext	An optional text appearing in a distinct node.
legendimage	An optional image file appearing in a distinct node.
qualityimage	An optional image file appearing in a distinct node.
showdepth	Logical. If TRUE, information about depth of the tree is added to the plot.
withquality	If TRUE, a node displaying fitting measures of the tree is added to the plot.
quality.fontsize	
	Numeric. Size of the font of the fitting measures node.
seqdata	a sequence object as defined by the the seqdef function.
sortv	The name of an optional variable used to sort the data before plotting, see seqplot.
dist.matrix	The name of an optional dissimilarity matrix used to find representative sequences, seqrplot.
withlegend	defines if and where the legend of the state colors is plotted. The default value "auto" sets the position of the legend automatically. Other possible value is "right". Obsolete value TRUE is equivalent to "auto".
legend.fontsize	
	Size of the font of the legend.
axes	if set to "all" (default value) x axes are drawn for each plot in the graphic. If set to "bottom" and group is used, axes are drawn only under the plots located at the bottom of the graphic area. If FALSE, no x axis is drawn.
...	other parameters that will be passed to imagefunc or seqplot (for seqtree2dot).

Details

These functions generate a "dot" file that can be used in GraphViz (http://www.graphviz.org). It also generates one image per node through a call to imagefunc passing the selected lines of imagedata if present or otherwise a list of indexes (of individuals belonging to a node). These functions are not intended to be used by end-user. See seqtreedisplay and disstreedisplay for a much simpler way to generate a graphical representation of a tree (seqtree or disstree).

seqtree2dot is a shortcut for sequences objects using the plot function seqplot. For each node, it calls seqplot with the corresponding subset of rows of seqdata and the provided seqplot's arguments. You should at least specify the type of the plot (e.g. type="d", see seqplot for more details).

If use.title is TRUE, imagefunc should take care to leave enough space for the title.

disstree2dotp is a simplified interface of disstree2dot which automatically leaves enough space for the title and subtitles. These functions are intended to be generic.

Value

Nothing but generates a "dot" and several image files (one per node) in the current working directory (see [getwd](#) and [setwd](#)).

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

[seqtree](#) and [seqtreedisplay](#), [disstree](#) and [disstreedisplay](#).

---

disstreeleaf   *Terminal node membership*

Description

Return a factor with the terminal node membership of each case.

Usage

```
disstreeleaf(tree, label=FALSE)
```

Arguments

tree   The tree, a `disstree` or `DissTreeNode` object.

label   Logical. If `TRUE`, the returned leaf memberships are labelled with the corresponding classifications rules.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

[disstree](#) for examples

---

dissvar   *Dissimilarity based discrepancy*

Description

Compute the discrepancy from the pairwise dissimilarities between objects. The discrepancy is a measure of dispersion of the set of objects.

Usage

```
dissvar(diss, weights=NULL, squared = FALSE)
```

Arguments

diss   A dissimilarity matrix or a `dist` object (see [dist](#))

weights   optional numerical vector containing weights.

squared   Logical. If `TRUE` diss is squared.

Details

The discrepancy is an extension of the concept of variance to any kind of objects for which we can compute pairwise dissimilarities. The discrepancy s^2 is defined as:

$$s^2 = \frac{1}{2n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}$$

Mathematical ground: In the Euclidean case, the sum of squares can be expressed as:

$$SS = \sum_{i=1}^{n} (y_i - \bar{y})^2 = \frac{1}{2n} \sum_{i=1}^{n} \sum_{j=1}^{n} (y_i - y_j)^2$$

The concept of discrepancy generalizes the equation by allowing to replace the $(y_i - y_j)^2$ term with any measure of dissimilarity d_{ij}.

Value

The discrepancy.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

References

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2011). Discrepancy analysis of state sequences, *Sociological Methods and Research*, Vol. 40(3), 471-510.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2010) Discrepancy analysis of complex objects using dissimilarities. In F. Guillet, G. Ritschard, D. A. Zighed and H. Briand (Eds.), *Advances in Knowledge Discovery and Management*, Studies in Computational Intelligence, Volume 292, pp. 3-19. Berlin: Springer.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2009) Analyse de dissimilarités par arbre d'induction. In EGC 2009, *Revue des Nouvelles Technologies de l'Information*, Vol. E-15, pp. 7-18.

Anderson, M. J. (2001) A new method for non-parametric multivariate analysis of variance. *Austral Ecology* **26**, 32-46.

Batagelj, V. (1988) Generalized ward and related clustering problems. In H. Bock (Ed.), *Classification and related methods of data analysis*, Amsterdam: North-Holland, pp. 67-74.

See Also

dissassoc to test association between objects represented by their dissimilarities and a covariate.
disstree for an induction tree analyse of objects characterized by a dissimilarity matrix.
disscenter to compute the distance of each object to its group center from pairwise dissimilarities.
dissmfac to perform multi-factor analysis of variance from pairwise dissimilarities.

Examples

```
## Defining a state sequence object
data(mvad)
mvad.seq <- seqdef(mvad[, 17:86])

## Building dissimilarities (any dissimilarity measure can be used)
mvad.ham <- seqdist(mvad.seq, method="HAM")

## Pseudo variance of the sequences
print(dissvar(mvad.ham))
```

---

ex1   *Example data set with missing values and weights*

Description

Example data set used to demonstrate the handling of missing values and weights.

The state columns (variable) are named `'P1'` to `'P13'`.

The alphabet is made of four possible states: `A, B, C` and `D`.

The data set contains also case weights (variable `weights`). The sum of the weights is 60.

Usage

```
data(ex1)
```

Format

A data frame with 7 rows, 13 state variables, 1 weight variable.

Source

The brain of the TraMineR package team.

---

ex2   *Example data sets with weighted and unweighted sequence data*

Description

Example data sets used to demonstrate the handling of weights. The `'ex2.weighted'` data set contains 6 sequences with weights inflating to 100 sequences (sum of weights is 100). The second data frame `'ex2.unweighted'` contains the corresponding 100 sequences.

The sequences are, in both data frames, in the `'seq'` column, and weights in the `'weight'` column of `'ex2.weighted'`.

The alphabet is made of four possible states: `A`, `B`, `C` and `D`.

These data sets are mainly intended to test and illustrate the handling of weights in TraMineR's functions. Weighted results obtained with 'ex2.weighted' data set should be exactly the same as unweighted results obtained with the 'ex2.unweighted' data set.

Usage

```
data(ex2)
```

Format

The command `data(ex2)` generates two data frames:
ex2.weighted: a data frame with 6 rows, 1 variable containing sequences as character strings, 1 weight variable.
ex2.unweighted: a data frame with 100 rows, 1 variable containing sequences as character strings.

Source

The brain of the TraMineR package team.

Examples

```
data(ex2)

ex2w.seq <- seqdef(ex2.weighted, 1, weights=ex2.weighted$weight)
ex2u.seq <- seqdef(ex2.unweighted)
```

---

famform   *Example data set: sequences of family formation*

Description

This data set contains 5 sequences of family formation histories, used by Elzinga (2008) to intro-
duce several metrics for computing distances between sequences. These sequences don't contain
information about the duration spent in each state, they contain only distinct successive states.

Usage

```
data(famform)
```

Format

A data frame with 5 rows and 1 variable.

Details

The sequences are in 'STS' format and stored in character strings with states separated with '-'.

This data set is used in TraMineR's manual to crosscheck some results with those presented by
Elzinga.

Source

Elzinga (2008)

References

Elzinga, Cees H. (2008). Sequence analysis: Metric representations of categorical time series. Non
published manuscript. VU University, Amsterdam.

---

mvad   *Example data set: Transition from school to work*

Description

The data comes from a study by McVicar and Anyadike-Danes on transition from school to work.
The data consist of static background characteristics and a time series sequence of 72 monthly
labour market activities for each of 712 individuals in a cohort survey. The individuals were fol-
lowed up from July 1993 to June 1999. The monthly states are recorded in columns 15 (Jul.93) to
86 (Jun.99).

States are:

```
                              employment   (EM)
                              FE           further education (FE)
                              HE           higher education (HE)
                              joblessness  (JL)
                              school       (SC)
                              training     (TR)
```

The data set contains also ids (`id`) and sample weights (`weight`) as well as the following binary covariates:

`male`
`catholic`
`Belfast`, `N.Eastern`, `Southern`, `S.Eastern`, `Western` (location of school, one of five Education and Library Board areas in Northern Ireland)
`Grammar` (type of secondary education, 1=grammar school)
`funemp` (father's employment status at time of survey, 1=father unemployed)
`gcse5eq` (qualifications gained by the end of compulsory education, 1=5+ GCSEs at grades A-C, or equivalent)
`fmpr` (SOC code of father's current or most recent job, 1=SOC1 (professional, managerial or related))
`livboth` (living arrangements at time of first sweep of survey (June 1995), 1=living with both parents)

Usage

```
data(mvad)
```

Format

A data frame containing 712 rows, 72 state variables, 1 id variable and 13 covariates.

Source

McVicar and Anyadike-Danes (2002)

References

McVicar, Duncan and Anyadike-Danes, Michael (2002). Predicting Successful and Unsuccessful Transitions from School to Work by Using Sequence Methods, *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, 165, 2, pp. 317–334.

---

plot.seqdiff   *Plotting a seqdiff object.*

Description

Plot method for the sliding values returned by [seqdiff](). Plots a statistic (the Pseudo R2 by default) along the position axis.

Usage

```
## S3 method for class 'seqdiff'
plot(x, stat = "Pseudo R2", type = "l", ylab = stat, xlab = "",
     legendposition = "top", ylim = NULL,
     xaxt = TRUE, col = NULL, xtstep=NULL, ...)
```

Arguments

x	an object produced by [seqdiff]()
stat	character. Name of the statistic to be plotted. Can be any of the statistics returned by [seqdiff]() or "discrepancy". See details.
type	the line type, see [lines]()
ylab	character: y-axis label.
xlab	character: x-axis label.
legendposition	character: position of the line legend, see [legend]()
ylim	numeric: if not NULL, range of the y-axis.
xaxt	logical: if TRUE an x-axis is plotted.
col	list of colors to use for each line.
xtstep	integer: optional step between tick-marks and labels on the x-axis. If unspecified, the xtstep attribute of the sequence object x is used. (see [seqdef]())
...	Additional parameters passed to [lines]()

Details

The function plots the sliding values of the requested statistic.

You can plot the evolution of two statistics by providing for instance stat=c("Pseudo R2", "Levene").

Use stat="discrepancy" to plot the within-discrepancies.

For "discrepancy", a separate line is drawn for the whole set of sequences and for each group. Those two values cannot be paired with another statistic.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

seqdiff

---

plot.stslist   *Plot method for state sequence objects*

Description

This is the plot method for state sequence objects of class stslist created by the seqdef function. It produces a sequence index plot.

Usage

```
## S3 method for class 'stslist'
plot(x, tlim=NULL, weighted=TRUE, sortv=NULL,
    cpal=NULL, missing.color=NULL,
    ylab, yaxis = TRUE, xaxis = TRUE, ytlab = NULL, ylas=0,
    xtlab = NULL, xtstep = NULL, cex.plot=1, ...)
```

Arguments

x	A state sequence object created with the seqdef function.
tlim	Indexes of the sequences to be plotted (default value is 1:10), for instance 20:50 to plot sequences 20 to 50, c(2,8,12,25) to plot sequences 2,8,12 and 25 in seqdata. If set to 0, all sequences in seqdata are plotted.
weighted	Logical: Should the bar representing each sequence be proportional to its weight? Ignored when no weights are assigned to sequences (see seqdef.)
sortv	A sorting variable or a sort method (one of "from.start" or "from.end"). See details.
cpal	alternative color palette to use for the states. If user specified, a vector of colors with number of elements equal to the number of states in the alphabet. By default, the cpal attribute of the seqdata sequence object is used (see seqdef).
missing.color	alternative color for representing missing values inside the sequences. By default, this color is taken from the "missing.color" attribute of the x sequence object.
ylab	An optional label for the y axis. If set to NA, no label is drawn.
yaxis	Controls whether the y axis is plotted or not. When set to TRUE, sequence indexes are displayed.
xaxis	if TRUE (default), the x (time) axis is plotted.
ytlab	the labels of the plotted sequences to display on the y axis. Default is the indexes of the sequences as defined by the tlim argument. Can be set to "id" for displaying the row names (id) of the sequences instead of their indexes; row names can be assigned to the sequence object with the id argument of the seqdef function or afterwards with rownames. Otherwise ytlab can be set to a vector of length equal to the number of sequences to be plotted.

ylas	sets the orientation of the sequence labels appearing on the y axis. Accepted values are the same as for the las standard option
	0: always parallel to the axis (default),
	1: always horizontal,
	2: always perpendicular to the axis,
	3: always vertical.
xtlab	optional labels for the x axis ticks labels. If unspecified, the column names of the seqdata sequence object are used (see [seqdef](#)).
xtstep	optional interval at which the tick-marks and labels of the x-axis are displayed. For example, with xtstep=3 a tick-mark is drawn at position 1, 4, 7, etc... The display of the corresponding labels depends on the available space and is dealt with automatically. If unspecified, the xtstep attribute of the x object is used.
cex.plot	expansion factor for setting the size of the font for the axis labels and names of the axes. The default value is 1. Values lesser than 1 will reduce the size of the font, values greater than 1 will increase it.
...	arguments to be passed to the plot function or other graphical parameters.

Details

This is the default plot method for state sequence objects (produced by the [seqdef](#) function), i.e., for objects of class *stslist*. It produces a sequence index plot, where individual sequences are rendered with stacked bars depicting the states over time.

This method is called by the generic [seqplot](#) function (if type="i"). The latter produces more sophisticated plots, allowing grouping and automatic display of the state color legend. The seqiplot function is a shortcut for calling seqplot with type="i".

When a sortv variable is provided to seqiplot or seqIplot, its values define the order in which the sequences are plotted. With sortv = "from.start", sequence are sorted by the elements of the alphabet at the successive positions starting from the beginning of the sequences. The "from.end" method proceeds similarly, but backward from the last position.

The interest of sequence index plots has for instance been stressed by *Scherer (2001)* and *Brzinsky-Fay et al. (2006)*. Notice that such index plots for thousands of sequences result in very heavy graphic files if they are stored in PDF or POSTSCRIPT format. To reduce the size, we suggest saving the figures in bitmap format by using for instance [png](#) instead of [postscript](#) or [pdf](#).

See Also

[seqplot](#)

Examples

```
## Defining a sequence object with the data in columns 10 to 25
## (family status from age 15 to 30) in the biofam data set
data(biofam)
biofam.lab <- c("Parent", "Left", "Married", "Left+Marr",
"Child", "Left+Child", "Left+Marr+Child", "Divorced")
```

```
biofam.seq <- seqdef(biofam, 10:25, labels=biofam.lab)

## Plot of the 10 most frequent sequences
## with bar width proportional to the frequency
plot(biofam.seq)

## Plotting the all data set
## with no borders
plot(biofam.seq, tlim=0, space=0, border=NA)

## =======
## Weights
## =======
data(ex1)
ex1.seq <- seqdef(ex1, 1:13, weights=ex1$weights)
plot(ex1.seq)
plot(ex1.seq, weighted=FALSE)
```

---

  plot.stslist.freq   *Plot method for sequence frequency tables*

Description

Plot method for output produced by the seqtab function, i.e objects of class stslist.freq.

Usage

```
## S3 method for class 'stslist.freq'
plot(x, cpal = NULL, missing.color = NULL, pbarw = TRUE,
    ylab = NULL, yaxis = TRUE, xaxis = TRUE,
    xtlab = NULL, xtstep = NULL, cex.plot = 1, ...)
```

Arguments

x	an object of class stslist.freq as produced by the seqtab function.
cpal	alternative color palette to be used for the states. If user specified, a vector of colors with number of elements equal to the number of states in the alphabet. By default, the 'cpal' attribute of the x object is used.
missing.color	alternative color for representing missing values inside the sequences. By default, this color is taken from the missing.color attribute of the x object.
pbarw	if pbarw=TRUE (default), the width of the bars are proportional to the sequence frequency in the dataset.
ylab	an optional label for the y axis. If set to NA, no label is drawn.
yaxis	if TRUE or "cum", the y axis is plotted with a label showing the cumulated percentage frequency of the displayed sequences. If "pct", the percentage value for each sequence is displayed.
xaxis	if TRUE (default) the x-axis is plotted.

xtlab	optional labels for the ticks of the x-axis. If unspecified, the `names` attribute of the x object is used.
xtstep	optional interval at which the tick-marks and labels of the x-axis are displayed. For example, with xtstep=3 a tick-mark is drawn at position 1, 4, 7, etc... The display of the corresponding labels depends on the available space and is dealt with automatically. If unspecified, the `xtstep` attribute of the x object is used.
cex.plot	expansion factor for setting the size of the font for the axis labels and names. The default value is 1. Values smaller than 1 will reduce the size of the font, values greater than 1 will increase the size.
...	further graphical parameters. For example `border=NA` to remove the bars borders, `space=0` to remove space between sequences. For more details about the graphical parameter arguments, see `barplot` and `par`.

Details

This is the plot method for the output produced by the [seqtab](#) function, i.e. objects of class *stslist.freq*. It produces a plot showing the sequences sorted bottom up according to their frequency in the data set.

This method is called by the generic [seqplot](#) function (if type="f") that produces more sophisticated plots, allowing grouping and automatic display of the state color legend. The seqfplot function is a shortcut for calling seqplot with type="f".

Author(s)

Alexis Gabadinho

Examples

```
## Loading the 'actcal' example data set
data(actcal)

## Defining a sequence object with data in columns 13 to 24
## (activity status from january to december 2000)
actcal.lab <- c(">37 hours", "19-36 hours", "1-18 hours", "no work")
actcal.seq <- seqdef(actcal, 13:24, labels=actcal.lab)

## 10 most frequent sequences in the data
actcal.freq <- seqtab(actcal.seq)

## Plotting the object
plot(actcal.freq, main="Sequence frequencies - actcal data set")

## Plotting all the distinct sequences without borders
## and space between sequences
actcal.freq2 <- seqtab(actcal.seq, tlim=0)
plot(actcal.freq2, main="Sequence frequencies - actcal data set",
  border=NA, space=0)
```

---

plot.stslist.meant   *Plot method for objects produced by the seqmeant function*

Description

This is the plot method for objects of class *stslist.meant* produced by the [seqmeant](#) function.

Usage

```
## S3 method for class 'stslist.meant'
plot(x, cpal = NULL, ylab = NULL, yaxis = TRUE, xaxis = TRUE,
     cex.plot = 1, ylim = NULL, ...)
```

Arguments

x	an object of class stslist.meant as produced by the seqmeant function.
cpal	alternative color palette to use for the states. If user specified, a vector of colors with number of elements equal to the number of states in the alphabet. By default, the 'cpal' attribute of the 'seqdata' sequence object is used (see [seqdef](#)).
ylab	an optional label for the y axis. If set to NA, no label is drawn.
yaxis	controls whether the y axis is plotted. Default is TRUE.
xaxis	if TRUE (default) the xaxis is plotted.
cex.plot	expansion factor for setting the size of the font for the axis labels and names. The default value is 1. Values lesser than 1 will reduce the size of the font, values greater than 1 will increase the size.
ylim	an optional vector setting the limits for the y axis. If NULL (default), limits are set to (0, max. sequence length).
...	further graphical parameters. For more details about the graphical parameter arguments, see barplot and par.

Details

This is the plot method for the output produced by the [seqmeant](#) function, i.e., objects of class *stslist.meant*. It produces a plot showing the mean times spent in each state of the alphabet.

When the "se" attribute of x is TRUE, i.e., when x contains also the standard errors of the mean times, error bars are automatically displayed on the plot. See the serr argument of [seqmeant](#).

This method is called by the generic [seqplot](#) function (if type="mt") that produces more sophisticated plots, allowing grouping and automatic display of the states legend. The [seqmtplot](#) function is a shortcut for calling seqplot with type="mt".

Examples

```
## Loading the mvad data set and creating a sequence object
data(mvad)
mvad.labels <- c("employment", "further education", "higher education",
                 "joblessness", "school", "training")
mvad.scodes <- c("EM","FE","HE","JL","SC","TR")
mvad.seq <- seqdef(mvad, 15:86, states=mvad.scodes, labels=mvad.labels)

## Computing the mean times
mvad.meant <- seqmeant(mvad.seq)

## Plotting
plot(mvad.meant, main="Mean durations in each state of the alphabet")

## Changing the y axis limits
plot(mvad.meant, main="Mean durations in each state of the alphabet",
 ylim=c(0,40))

## Displaying error bars
plot(mvad.meant, main="Mean durations in each state of the alphabet",
 ylim=c(0,40), serr=TRUE)
```

---

plot.stslist.modst   *Plot method for modal state sequences*

Description

Plot method for output produced by the seqmodst function, i.e objects of class stslist.modst.

Usage

```
## S3 method for class 'stslist.modst'
plot(x, cpal = NULL, ylab = NULL, yaxis = TRUE, xaxis = TRUE,
     xtlab = NULL, xtstep = NULL, cex.plot = 1, ...)
```

Arguments

x	an object of class stslist.modst as produced by the seqmodst function.
cpal	alternative color palette to use for the states. If user specified, a vector of colors with number of elements equal to the number of states in the alphabet. By default, the 'cpal' attribute of the x object is used.
ylab	an optional label for the y axis. If set to NA, no label is drawn.
yaxis	if TRUE (default) the y axis is plotted.
xaxis	if TRUE (default) the x axis is plotted.
xtlab	optional labels for the x axis ticks. If unspecified, the names attribute of the x object is used.

xtstep optional interval at which the tick-marks and labels of the x-axis are displayed. For example, with xtstep=3 a tick-mark is drawn at position 1, 4, 7, etc... The display of the corresponding labels depends on the available space and is dealt with automatically. If unspecified, the xtstep attribute of the x object is used.

cex.plot expansion factor for setting the size of the font for the axis labels and names. The default value is 1. Values lesser than 1 will reduce the size of the font, values greater than 1 will increase the size.

... further graphical parameters. For more details about the graphical parameter arguments, see barplot and par.

Details

This is the plot method for the output produced by the [seqmodst](#) function, i.e. objects of class *stslist.modst*. It produces a plot showing the sequence of modal states with bar width proportional to the state frequencies.

This method is called by the generic [seqplot](#) function (if type="ms") that produces more sophisticated plots, allowing grouping and automatic display of the states legend. The seqmsplot function is a shortcut for calling seqplot with type="ms".

Examples

```
## Defining a sequence object with the data in columns 10 to 25
## (family status from age 15 to 30) in the biofam data set
data(biofam)
biofam.lab <- c("Parent", "Left", "Married", "Left+Marr",
"Child", "Left+Child", "Left+Marr+Child", "Divorced")
biofam.seq <- seqdef(biofam, 10:25, labels=biofam.lab)

## Modal state sequence
biofam.modst <- seqmodst(biofam.seq)
plot(biofam.modst)
```

---

plot.stslist.rep *Plot method for representative sequence sets*

Description

This is the plot method for output produced by the [seqrep](#) function, i.e, for objects of class *stslist.rep*. It produces a representative sequence plot.

Usage

```
## S3 method for class 'stslist.rep'
plot(x, cpal = NULL, missing.color=NULL, pbarw = TRUE,
    dmax = NULL, stats=TRUE, ylab = NULL, xaxis = TRUE,
    xtlab = NULL, xtstep = NULL, cex.plot = 1, ...)
```

Arguments

x	an object of class *stslist.rep* as produced by the seqrep function.
cpal	alternative color palette to use for the states. If user specified, a vector of colors with number of elements equal to the number of states in the alphabet. By default, the 'cpal' attribute of the x object is used.
missing.color	alternative color for representing missing values inside the sequences. By default, this color is taken from the "missing.color" attribute of the sequence object being plotted.
pbarw	when TRUE, the bar heights are set proportional to the number of represented sequences.
dmax	maximal theoretical distance, used for the x axis limits.
stats	if TRUE (default), mean discrepancy in each subset defined by all sequences attributed to one representative sequence and the mean distance to this representative sequence are displayed.
ylab	an optional label for the y axis. If set to NA, no label is drawn.
xaxis	controls whether a x axis is plotted.
xtlab	optional labels for the x axis ticks labels. If unspecified, the column names of the object being plotted.
xtstep	optional interval at which the tick-marks and labels of the x-axis are displayed. For example, with xtstep=3 a tick-mark is drawn at position 1, 4, 7, etc... The display of the corresponding labels depends on the available space and is dealt with automatically. If unspecified, the xtstep attribute of the x object is used.
cex.plot	expansion factor for setting the size of the font for the axis labels and names. The default value is 1. Values lesser than 1 will reduce the size of the font, values greater than 1 will increase the size.
...	further graphical parameters. For more details about the graphical parameter arguments, see barplot and par.

Details

This is the plot method for the output produced by the seqrep function, i.e. objects of class *stslist.rep*. It produces a plot where the representative sequences are displayed as horizontal bars with width proportional to the number of sequences assigned to them. Sequences are plotted bottom-up according to their representativeness score.

Above the plot, two parallel series of symbols associated to each representative are displayed horizontally on a scale ranging from 0 to the maximal theoretical distance D_{max}. The location of the symbol associated to the representative r_i indicates on axis A the (pseudo) variance (V_i) within the subset of sequences assigned to r_i and on the axis B the mean distance MD_i to the representative.

This method is called by the generic seqplot function (if type="r") that produces more sophisticated plots with group splits and automatic display of the color legend. The seqrplot function is a shortcut for calling seqplot with type="r".

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

Examples

```
## Loading the mvad data set and creating a sequence object
data(mvad)
mvad.labels <- c("employment", "further education", "higher education",
                 "joblessness", "school", "training")
mvad.scodes <- c("EM","FE","HE","JL","SC","TR")

## First 36 months trajectories
mvad.seq <- seqdef(mvad, 15:50, states=mvad.scodes, labels=mvad.labels)

## Computing Hamming distances
##
dist.ham <- seqdist(mvad.seq, method="HAM")

## Extracting a representative set using the sequence frequency
## as a representativeness criterion
mvad.rep <- seqrep(mvad.seq, dist.matrix=dist.ham)

## Plotting the representative set
plot(mvad.rep)
```

---

plot.stslist.statd   *Plot method for objects produced by the seqstatd function*

Description

This is the plot method for output produced by the [seqstatd](#) function, i.e for objects of class *stslist.statd*.

Usage

```
## S3 method for class 'stslist.statd'
plot(x, type = "d", cpal = NULL, ylab = NULL,
     yaxis = TRUE, xaxis = TRUE, xtlab = NULL, xtstep = NULL, cex.plot = 1,
     space=0, ...)
```

Arguments

x	an object of class stslist.statd as produced by the [seqstatd](#) function.
type	if "d" (default), a state distribution plot is produced. If "Ht" an entropy index plot is produced.
cpal	alternative color palette to be used for the states. If user specified, a vector of colors with number of elements equal to the number of states in the alphabet. By default, the 'cpal' attribute of the x object is used.
ylab	an optional label for the y axis. If set to NA, no label is drawn.
yaxis	if TRUE or "cum", the y axis is plotted with a label showing the cumulated percentage frequency of the displayed sequences. If "pct", the percentage value for each sequence is displayed.

xaxis	if TRUE (default) the x-axis is plotted.
xtlab	optional labels for the ticks of the x-axis. If unspecified, the names attribute of the input x object is used.
xtstep	optional interval at which the tick-marks and labels of the x-axis are displayed. For example, with xtstep=3 a tick-mark is drawn at position 1, 4, 7, etc... The display of the corresponding labels depends on the available space and is dealt with automatically. If unspecified, the xtstep attribute of the x object is used.
cex.plot	expansion factor for setting the size of the font for the axis labels and names. The default value is 1. Values smaller than 1 will reduce the size of the font, values greater than 1 will increase the size.
space	the space between the stacked bars. Default is 0, i.e. no space.
...	further graphical parameters such as border=NA to remove the borders of the bars. For more details about the graphical parameter arguments, see barplot and par.

Details

This is the plot method for the output produced by the [seqstatd](#) function, i.e. for objects of class *stslist.statd*. If type="d" it produces a state distribution plot presenting the sequence of the transversal state frequencies at each successive (time) position, as computed by the [seqstatd](#) function. With type="Ht", the series of entropies of the transversal state distributions is plotted.

This method is called by the generic [seqplot](#) function (if type="d" or type="Ht") that produces more sophisticated plots, allowing grouping and automatic display of the state color legend. The seqdplot and seqHtplot functions are shortcuts for calling seqplot with type="d" or type="Ht" respectively.

Examples

```
## Defining a sequence object with the data in columns 10 to 25
## (family status from age 15 to 30) in the biofam data set
data(biofam)
biofam.lab <- c("Parent", "Left", "Married", "Left+Marr",
"Child", "Left+Child", "Left+Marr+Child", "Divorced")
biofam.seq <- seqdef(biofam, 10:25, labels=biofam.lab)

## State distribution
biofam.statd <- seqstatd(biofam.seq)

## State distribution plot (default type="d" option)
plot(biofam.statd)

## Entropy index plot
plot(biofam.statd, type="Ht")
```

plot.subseqelist	*Plot frequencies of subsequences*

Description

Plot frequencies of subsequences.

Usage

```
## S3 method for class 'subseqelist'
plot(x, freq=NULL,cex=1,...)
```

Arguments

x	The subsequences to plot (a subseqelist object
freq	The frequencies to plot, support if NULL
cex	Font size. See par.
...	arguments passed to barplot

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

seqefsub

Examples

```
## loading data
data(actcal.tse)

## creating sequences
actcal.seqe <- seqecreate(actcal.tse)

## Looking for frequent subsequences
fsubseq <- seqefsub(actcal.seqe,pMinSupport=0.01)

## Frequence of first ten subsequences
plot(fsubseq[1:10], cex=2)
plot(fsubseq[1:10])
```

plot.subseqelistchisq   *Plot discriminant subsequences*

Description

Plot the result of [seqecmpgroup](#)

Usage

```
## S3 method for class 'subseqelistchisq'
plot(x, ylim = "uniform", rows = NA, cols = NA,
    residlevels = c(0.05,0.01),
    cpal = brewer.pal(1 + 2 * length(residlevels), "RdBu"),
    legendcol = NULL, legend.cex = 1, ptype="freq",
    legend.title = NULL, ...)
```

Arguments

x	The subsequences to plot (a subseqelist object).
ylim	if "uniform" all axes have same limits.
rows	Number of graphic rows
cols	Number of graphic columns
residlevels	Significance levels used to colorize the Pearson residual
cpal	Color palette used to color the results
legendcol	When TRUE the legend is printed vertically, when FALSE it is printed horizontally. If NULL (default) the best position will be chosen.
legend.cex	Scale parameters for text legend.
ptype	If set to "resid", Pearson residuals are plotted instead of frequencies
legend.title	Legend title.
...	Additional parameters passed to [barplot](#)

Value

nothing

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

[seqecmpgroup](#)

---

read.tda.mdist *Read a distance matrix produced by TDA.*

Description

This function reads a distance matrix produced by TDA into an R object. When computing OM distances in TDA, the output is a 'half' matrix stored in a text file as a vector.

Usage

```
read.tda.mdist(file)
```

Arguments

file   the path to the file containing TDA output.

Value

a R matrix containing the distances.

---

seqalign *Computation details about a pairwise alignment*

Description

The function provides details about a pairwise alignment.

Usage

```
seqalign(seqdata, indices, indel=1, sm, with.missing = FALSE)

## S3 method for class 'seqalign'
plot(x, cpal=NULL, missing.color=NULL,
    ylab=NULL, yaxis=TRUE, xaxis=TRUE, ytlab=NULL, ylas=0,
    xtlab=NULL, cex.plot=1, ...)

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

Arguments

seqdata	a state sequence object defined with the [seqdef](#) function.
indices	a vector of length 2 giving the indexes of the two sequences
indel	indel cost (see [seqdist](#))
sm	matrix of substitution costs or a method for computing the costs (see [seqdist](#))
with.missing	logical: Should the missing state be considered as an element of the alphabet?
x	an object of class seqalign
cpal	color palette
missing.color	color for missing elements
ylab	y label
yaxis	yaxis
xaxis	xaxis
ytlab	ytlab
ylas	ylas
xtlab	xtlab
cex.plot	plot font size
digits	number of digits for printed output
...	additional arguments passed to other functions

Details

There are print and plot methods for seqalign objects.

Value

Object of class seqalign

Author(s)

Alexis Gabadinho (plot.seqalign) and Matthias Studer (seqalign) (with Gilbert Ritschard for the help page)

See Also

[seqdist](#)

Examples

```
data(biofam)
biofam.seq <- seqdef(biofam, 10:25)
costs <- seqsubm(biofam.seq, method="TRATE")
sa <- seqalign(biofam.seq, 1:2, indel=1, sm=costs)
print(sa)
plot(sa)
sa <- seqalign(biofam.seq, c(1,5), indel=0.5, sm=costs)
print(sa)
plot(sa)
```

---

seqcomp   *Compare two state sequences*

Description

Check whether two state sequences are identical.

Usage

```
seqcomp(x, y)
```

Arguments

x	a state sequence object containing a single sequence (typically the row of a main sequence object, see [seqdef](#))
y	a state sequence object containing a single sequence (typically the row of a main sequence object, see [seqdef](#))

Value

TRUE if sequences are identical, FALSE otherwise

See Also

[seqfind](#), [seqfpos](#), [seqpm](#)

Examples

```
data(mvad)
mvad.shortlab <- c("EM", "FE", "HE", "JL", "SC", "TR")
mvad.seq <- seqdef(mvad, states=mvad.shortlab, 15:86)

## Comparing sequences 1 and 2 in mvad.seq
seqcomp(mvad.seq[1,],mvad.seq[2,])

## Comparing sequences 176 and 211 in mvad.seq
seqcomp(mvad.seq[176,],mvad.seq[211,])
```

seqconc   *Concatenate vectors of states or events into a character string*

Description

Concatenate vectors of states or events into a character string. In the string, each state is separated by 'sep'. The void elements in the input sequences are eliminated.

Usage

```
seqconc(data, var=NULL, sep="-", vname="Sequence", void=NA)
```

Arguments

data   A dataframe or matrix containing sequence data.

var   List of the columns containing the sequences. Default is NULL in which case all
              columns are retained. Whether the sequences are in the compressed (character
              strings) or extended format is automatically detected by counting the number of
              columns.

sep   Character used as separator. By default, "-".

vname   an optional name for the variable containing the sequences. By default, "Sequence".

void   the code used for void elements appearing in the sequences (see *Gabadinho et
              al. (2009)* for more details on missing values and void elements in sequences).
              Default is NA.

Value

a vector of character strings, one for each row in the input data.

Author(s)

Alexis Gabadinho

References

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2009). Mining Sequence Data in R with
the TraMineR package: A user's guide. *Department of Econometrics and Laboratory of Demography, University of Geneva.*

See Also

[seqdecomp](#).

Examples

```
data(actcal)
actcal.string <- seqconc(actcal,13:24)
head(actcal.string)
```

---

seqdecomp   *Convert a character string into a vector of states or events*

Description

For the moment, each character in the string will be considered to be one state or event = this function will not give accurate results if the character string representing the sequence contains events or states coded with more than one character.

Usage

```
seqdecomp(data, var=NULL, sep='-', miss="NA", vnames=NULL)
```

Arguments

data   a dataframe or matrix containing sequence data.

var   the list of columns containing the sequences. Default is NULL, ie all the columns. Whether the sequences are in the compressed (character strings) or extended format is automatically detected by counting the number of columns.

sep   the between states/events separator used in the input data set. Default is '-'.

miss   the symbol for missing values (if any) used in the input data set. Default is NA.

vnames   optional names for the column/variables of the output data set. Default is NULL.

See Also

[seqconc](#).

Examples

```
## Converts 'seq' into a vector of states of length 10
seq <- "A-A-A-A-B-B-B-C-C-C"
seqdecomp(seq)
```

---

seqdef   *Create a state sequence object*

Description

Create a state sequence object with attributes such as alphabet, color palette and state labels. Most TraMineR functions for state sequences require such a state sequence object as input argument. There are specific methods for plotting, summarizing and printing state sequence objects.

Usage

```
seqdef(data, var=NULL, informat="STS", stsep=NULL,
       alphabet=NULL, states=NULL, id=NULL, weights=NULL, start=1,
       left=NA, right="DEL", gaps=NA, missing=NA, void="%", nr="*",
       cnames=NULL, xtstep=1, cpal=NULL, missing.color="darkgrey",
       labels=NULL, ...)
```

Arguments

data	a data frame or matrix containing sequence data.
var	the list of columns containing the sequences. Default is NULL, i.e. all the columns. The function detects automatically whether the sequences are in the compressed (successive states in a character string) or extended format.
informat	format of the original data. Default is "STS". Other available formats are: "SPS" and "SPELL", in which case the [seqformat](#) function is called to convert the data into the "STS" format (see TraMineR user's manual (*Gabadinho et al., 2010*) for a description of these formats). A better solution is nonetheless to convert first your data with [seqformat](#), so as to have better control over the conversion process and visualize the intermediate "STS" formatted data.
stsep	the character used as separator in the original data if input format is successive states in a character string. If NULL (default value), the [seqfcheck](#) function is called for detecting automatically a separator among "-" and ":". Other separators must be specified explicitly.
alphabet	optional vector containing the alphabet (the list of all possible states). Use this option if some states in the alphabet don't appear in the data or if you want to reorder the states. The specified vector MUST contain AT LEAST all the states appearing in the data. It may possibly contain additional states not appearing in the data. If NULL, the alphabet is set to the distinct states appearing in the data as returned by the [seqstatl](#) function. See details.
states	an optional vector containing the short state labels. Must have a length equal to the size of the alphabet and the labels must be ordered conformably with alpha-numeric ordered values returned by the [seqstatl](#) function, or, when alphabet= is set, with the thus newly defined alphabet.
id	optional argument for setting the rownames of the sequence object. If NULL (default), the rownames are taken from the input data. If set to "auto", sequences are numbered from 1 to the number of sequences. A vector of rownames of length equal to the number of sequences may be specified as well.
weights	optional numerical vector containing weights, which are accounted for by plotting and statistical functions when applicable.
start	starting time. For instance, if sequences begin at age 15, you can specify 15. At this stage, used only for labelling column names.
left	the behavior for missing values appearing before the first (leftmost) valid state in each sequence. See *Gabadinho et al. (2010)* for more details on the options for handling missing values when defining sequence objects. By default, left missing values are treated as 'real' missing values and converted to the internal

	missing value code defined by the nr option. Other options are "DEL" to delete the positions containing missing values or a state code (belonging to the alphabet or not) to replace the missing values.
right	the behavior for missing values appearing after the last (rightmost) valid state in each sequence. Same options as for the left argument.
gaps	the behavior for missing values appearing inside the sequences, i.e. after the first (leftmost) valid state and before the last (rightmost) valid state of each sequence. Same options as for the left argument.
missing	the code used for missing values in the input data. When specified, all cells containing this value will be replaced by NA's, the internal R code for missing values. If 'missing' is not specified, cells containing NA's are considered as missing values.
void	the internal code used by TraMineR for representing void elements in the sequences. Default is"%".
nr	the internal code used by TraMineR for representing real missing elements in the sequences. Default is"*".
cnames	optional names for the columns composing the sequence data. Those names will be used by default in the graphics as axis labels. If NULL (default), names are taken from the original column names in the data.
xtstep	step between displayed tick-marks and labels on the x-axis of state sequence plots. If not overridden by the user, plotting functions retrieve this parameter from the xtstep attribute of the sequence object. For example, with xtstep=3 a tick-mark is displayed at positions 1, 4, 7, etc... Default value is 1; i.e., a tick mark is displayed at each position. The display of the corresponding labels depends on the available space and is dealt with automatically.
cpal	an optional color palette for representing the states in the graphics. If NULL (default), a color palette is created by calling the brewer.pal function of the RColorBrewer package. If number of states is less or equal than 8, the "Accent" palette is used. If number of states is between 8 and 12, the "Set3" palette is used. If the number of states in the data is greater than 12 you have to specify your own palette. The list of available colors is displayed by the colors function. You can also use alternatively some other palettes from the RColorBrewer package.
missing.color	alternative color for representing missing values inside the sequences. Defaults to "darkgrey".
labels	optional state labels used for the color legend of TraMineR's graphics. If NULL (default), the state names in the alphabet are used as state labels as well.
...	options passed to the seqformat function for handling input data that is not in STS format.

Details

Applying subscripts to sequence objects (eg. seq[,1:5] or seq[1:10,]) returns a state sequence object with some attributes preserved (alphabet, missing) and some others (start, column names) adapted to the selected column or row subset. If only one column is specified, a factor is returned.

For reordering the states use the `alphabet` argument. This may for instance be of interest when you want to compare data from different sources with different codings of similar states. Using `alphabet` permits to order the states conformably in all sequence objects. Otherwise, the default state order is the alpha-numeric order returned by the seqstatl function which may differ when you have different original codings.

Value

An object of class `stslist`. There are `print`, `plot` and `summary` methods for such objects. State sequence objects are required as argument to other functions such as plotting functions (`seqdplot`, `seqiplot` or `seqfplot`), functions to compute distances (`seqdist`), etc...

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for help page)

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2010). Mining Sequence Data in R with the TraMineR package: A user's guide. *Department of Econometrics and Laboratory of Demography, University of Geneva*.

See Also

plot.stslist to plot state sequence objects,
seqplot for high level plots of state sequence objects,
seqecreate to create an event sequence object,
seqformat for converting between various longitudinal data formats.

Examples

```
## Creating a sequence object with the columns 13 to 24
## in the 'actcal' example data set
data(actcal)
actcal.seq <- seqdef(actcal,13:24,
labels=c("> 37 hours", "19-36 hours", "1-18 hours", "no work"))

## Displaying the first 10 rows of the sequence object
actcal.seq[1:10,]

## Displaying the first 10 rows of the sequence object
## in SPS format
print(actcal.seq[1:10,], format="SPS")

## Plotting the first 10 sequences
plot(actcal.seq)

## Re-ordering the alphabet
actcal.seq <- seqdef(actcal,13:24,alphabet=c("B","A","D","C"))
```

```
alphabet(actcal.seq)

## Adding a state not appearing in the data to the
## alphabet
actcal.seq <- seqdef(actcal,13:24,alphabet=c("A","B","C","D","E"))
alphabet(actcal.seq)

## Adding a state not appearing in the data to the
## alphabet and changing the states labels
actcal.seq <- seqdef(actcal,13:24,
  alphabet=c("A","B","C","D","E"),
  states=c("FT","PT","LT","NO","TR"))
alphabet(actcal.seq)
actcal.seq[1:10,]

## ===========================
## Example with missing values
## ===========================
data(ex1)

## With right="DEL" default value
seqdef(ex1,1:13)

## Eliminating 'left' missing values
seqdef(ex1,1:13, left="DEL")

## Eliminating 'left' missing values and gaps
seqdef(ex1,1:13, left="DEL", gaps="DEL")

## ===================
## Example with weights
## ===================
ex1.seq <- seqdef(ex1, 1:13, weights=ex1$weights)

## weighted sequence frequencies
seqtab(ex1.seq)
```

---

seqdiff *Position-wise discrepancy analysis between groups of sequences*

Description

The function analyses how the differences between groups of sequences evolve along the positions. It runs a sequence of discrepancy analyses on sliding windows.

Usage

```
seqdiff(seqdata, group, cmprange = c(0, 1),
    seqdist_arg=list(method="LCS",norm=TRUE),
    with.missing = FALSE, weighted = TRUE, squared = FALSE)
```

Arguments

seqdata	a state sequence object created with the [seqdef](#) function.
group	The group variable.
cmprange	The time range of the sliding window on which subsequences are compared.
seqdist_arg	List of arguments passed to [seqdist](#) for computing the distances.
with.missing	Logical. If TRUE, missing values are considered as an additional state. If FALSE subsequences with missing values are removed from the analysis.
weighted	Logical. If TRUE, seqdiff uses the weights specified in seqdata.
squared	Logical. If TRUE the dissimilarities are squared for computing the discrepancy.

Details

The function analyses how the part of discrepancy explained by the group variable evolves along the position axis. It runs successively discrepancy analyses within a sliding time-window of range cmprange). At each position, the method uses [seqdist](#) to compute a distance matrix over the time-window and then derives the explained discrepancy on that window with [dissassoc](#).

There are print and plot methods for the returned value.

Value

A seqdiff object, with the following items:

stat	A data.frame with three statistics (PseudoF, PseudoR2 and PseudoT) for each time stamp of the sequence, see [dissassoc](#)
discrepancy	A data.frame with, at each time stamp, the discrepancy within each group defined by the group variable and for the whole population.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

References

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2010) Discrepancy analysis of complex objects using dissimilarities. In F. Guillet, G. Ritschard, D. A. Zighed and H. Briand (Eds.), *Advances in Knowledge Discovery and Management*, Studies in Computational Intelligence, Volume 292, pp. 3-19. Berlin: Springer.

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2009) Analyse de dissimilarités par arbre d'induction. In EGC 2009, *Revue des Nouvelles Technologies de l'Information*, Vol. E-15, pp. 7-18.

See Also

[dissassoc](#) to analyse the association of the group variable with the whole sequence

Examples

```
## Define a state sequence object
data(mvad)
## First 24 months trajectories
mvad.seq <- seqdef(mvad[, 17:40])

## Position-wise discrepancy analysis
mvad.diff <- seqdiff(mvad.seq, group=mvad$gcse5eq)
print(mvad.diff)
plot(mvad.diff, stat=c("Pseudo R2", "Levene"), xtstep=6)
plot(mvad.diff, stat="discrepancy")
```

---

seqdim   *Dimension of a set of sequences*

Description

Returns the number of sequences (rows) and the maximum length of a set of sequences.

Usage

```
seqdim(seqdata)
```

Arguments

seqdata   a set of sequences.

Details

The function will first search for separators '-' or ':' in the sequences in order to detect wether they are in the compressed or extended format.

Value

a vector with the number of sequences and the maximum sequence length.

Author(s)

Alexis Gabadinho

---

seqdist   *Distances (dissimilarities) between sequences*

Description

Computes pairwise dissimilarities between sequences or dissimilarities with a reference sequence. Several dissimilarities measures or metrics are available: optimal matching (OM), distance based on the longest common prefix (LCP), on the longest common suffix (RLCP), on the longest common subsequence (LCS), the Hamming distance (HAM) and the Dynamic Hamming Distance (DHD).

Usage

```
seqdist(seqdata, method, refseq=NULL, norm=FALSE,
     indel=1, sm=NA, with.missing=FALSE, full.matrix=TRUE)
```

Arguments

seqdata	a state sequence object defined with the [seqdef](#) function.
method	a character string indicating the metric to be used. One of ″OM″ (Optimal Matching), ″LCP″ (Longest Common Prefix), ″RLCP″ (reversed LCP, i.e. Longest Common Suffix), ″LCS″ (Longest Common Subsequence), ″HAM″ (Hamming distance), ″DHD″ (Dynamic Hamming distance).
refseq	Optional baseline sequence to compute the distances from. Can be the index of a sequence in the state sequence object, 0 for the most frequent sequence, or an external sequence passed as a sequence object with 1 row and same alphabet as seqdata assigned to it.
norm	if TRUE, the computed OM, LCP, RLCP or LCS distances are normalized to account for differences in sequence lengths, and the normalization method is automatically selected. Default is FALSE. Can also be one of ″none″, ″maxlength″, ″gmean″, ″maxdist″, ″YujianBo″. See details.
indel	the insertion/deletion cost (OM method). Default is 1. Ignored with non OM metrics.
sm	substitution-cost matrix (OM, HAM and DHD method). Can also be one of the [seqsubm](#) build methods ″TRATE″ or ″CONSTANT″. Default is NA. Ignored with LCP, RLCP and LCS metrics. A valid non NA value must be given for OM.
with.missing	must be set to TRUE when sequences contain non deleted gaps (missing values). See details.
full.matrix	If TRUE (default), the full distance matrix is returned. This is for compatibility with earlier versions of the seqdist function. If FALSE, an object of class [dist](#) is returned, that is, a vector containing only values from the upper triangle of the distance matrix. Since the distance matrix is symmetrical, no information is lost with this representation while size is divided by 2. Objects of class dist can be passed directly as arguments to most clustering functions. Ignored when refseq is set.

Details

The seqdist function returns a matrix of distances between sequences or a vector of distances to a reference sequence. The available metrics (see 'method' option) are optimal matching (″OM″), longest common prefix (″LCP″), longest common suffix (″RLCP″), longest common subsequence (″LCS″), Hamming distance (″HAM″) and Dynamic Hamming Distance (″DHD″). The Hamming distance is OM without indels and the Dynamic Hamming Distance is HAM with specific substitution costs at each position as proposed by *Lesnard (2006)*. Note that HAM and DHD apply only to sequences of equal length.

For OM, HAM and DHD, a user specified substitution cost matrix can be provided with the sm argument. For DHD, this should be a series of matrices grouped in a 3-dimensional matrix with the third index referring to the position in the sequence. When sm is not specified, a constant substitution cost of 1 is used with HAM, and *Lesnard (2006)*'s proposal for DHD.

Distances can optionally be normalized by means of the norm argument. If set to TRUE, Elzinga's normalization (similarity divided by geometrical mean of the two sequence lengths) is applied to LCP, RLCP and LCS distances, while Abbott's normalization (distance divided by length of the longer sequence) is used for OM, HAM and DHD. Elzinga's method can be forced with ″gmean″ and Abbott's rule with ″maxlength″. With ″maxdist″ the distance is normalized by its maximal possible value. For more details, see *Elzinga (2008)* and *Gabadinho et al. (2009)*.

When sequences contain gaps and the gaps=NA option was passed to [seqdef](), i.e. when there are non deleted missing values, the with.missing argument should be set to TRUE. If left to FALSE the function stops when it encounters a gap. This is to make the user aware that there are gaps in his sequences. If the OM method is selected, seqdist expects a substitution cost matrix with a row and a column entry for the missing state (symbol defined with the nr option of [seqdef]()). This will be the case for substitution cost matrices returned by [seqsubm](). More details on how to compute distances with sequences containing gaps are given in *Gabadinho et al. (2009)*.

Value

When refseq is specified, a vector with distances between the sequences in the data sequence object and the reference sequence is returned. When refseq is NULL (default), the whole matrix of pairwise distances between sequences is returned.

Author(s)

Matthias Studer, Alexis Gabadinho and Nicolas S. Müller (first version) (with Gilbert Ritschard for the help page)

References

Elzinga, Cees H. (2008). Sequence analysis: Metric representations of categorical time series. *Technical Report*, Department of Social Science Research Methods, Vrije Universiteit, Amsterdam.

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2009). Mining Sequence Data in R with the TraMineR package: A user's guide. Department of Econometrics and Laboratory of Demography, University of Geneva

Lesnard, L. (2006). Optimal Matching and Social Sciences. *Série des Documents de Travail du CREST*, Institut National de la Statistique et des Etudes Economiques, **2006-01**, Paris.

Studer, M. and G. Ritschard (2015). What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures. *Journal of the Royal Statistical Society A*, (Early view), DOI: 10.1111/rssa.12125.

See Also

seqsubm, seqdef, and for multichannel distances seqdistmc.

For more dissimilarity measures, consider the package *seqdist2* (available from R-Forge https://r-forge.r-project.org/R/?group_id=743) that proposes all the measures addressed in *Studer and Ritschard (2015)*.

Examples

```
## optimal matching distances with substitution cost matrix
## derived from transition rates
data(biofam)
biofam.seq <- seqdef(biofam, 10:25)
costs <- seqsubm(biofam.seq, method="TRATE")
biofam.om <- seqdist(biofam.seq, method="OM", indel=3, sm=costs)

## normalized LCP distances
biofam.lcp <- seqdist(biofam.seq, method="LCP", norm=TRUE)

## normalized LCS distances to the most frequent sequence in the data set
biofam.lcs <- seqdist(biofam.seq, method="LCS", refseq=0, norm=TRUE)

## histogram of the normalized LCS distances
hist(biofam.lcs)

## distance to an external sequence
refs <- seqdef("(0,5)-(3,5)-(4,6)", informat="SPS",
            alphabet=alphabet(biofam.seq))
biofam.ref <- seqdist(biofam.seq, method="LCS", refseq=refs)
hist(biofam.ref)

## ====================
## Example with missings
## ====================
data(ex1)
ex1.seq <- seqdef(ex1,1:13)

subm <- seqsubm(ex1.seq, method="TRATE", with.missing=TRUE)
ex1.om <- seqdist(ex1.seq, method="OM", sm=subm, with.missing=TRUE)
```

seqdistmc | *Multichannel distances between sequences*

Description

Compute multichannel pairwise distances between sequences. Several metrics are available: optimal matching (OM), the longest common subsequence (LCS), the Hamming distance (HAM) and the Dynamic Hamming Distance (DHD).

Usage

```
seqdistmc(channels, method, norm=FALSE, indel=1, sm=NULL,
    with.missing=FALSE, full.matrix=TRUE, link="sum", cval=2,
    miss.cost=2, cweight=NULL)
```

Arguments

channels	A list of state sequence objects defined with the [seqdef](#) function, each state sequence object corresponding to a "channel".
method	a character string indicating the metric to be used. One of ″OM″ (Optimal Matching), ″LCS″ (Longest Common Subsequence), ″HAM″ (Hamming distance), ″DHD″ (Dynamic Hamming distance).
norm	if TRUE, the computed distances are normalized to account for differences in sequence lengths. Default is FALSE. See details.
indel	A vector with an insertion/deletion cost for each channel (OM method).
sm	A list with a substitution-cost matrix for each channel (OM, HAM and DHD method) or a list of method names for generating the substitution-costs (see [seqsubm](#)).
with.missing	Must be set to TRUE when sequences contain non deleted gaps (missing values) or when channels are of different length. See details.
full.matrix	If TRUE (default), the full distance matrix is returned. If FALSE, an object of class [dist](#) is returned.
link	One of ″sum″ or ″mean″. Method to compute the "link" between channels. Default is to sum the substitution costs.
cval	Substitution cost for ″CONSTANT″ matrix, see [seqsubm](#).
miss.cost	Missing values substitution cost, see [seqsubm](#).
cweight	A vector of channel weights. Default is 1 (same weight for each channel).

Details

The seqdistmc function returns a matrix of multichannel distances between sequences. The available metrics (see 'method' option) are optimal matching (″OM″), longest common subsequence (″LCS″), Hamming distance (″HAM″) and Dynamic Hamming Distance (″DHD″). See [seqdist](#) for

more information about distances between sequences. The seqdistmc function computes a multichannel distance in two steps following the strategy proposed by *Pollock (2007)*. First it builds a new sequence object derived from the combination of the sequences of each channel. Second, it derives the substitution cost matrix by summing (or averaging) the costs of substitution across channels. It then calls seqdist to compute the final matrix. Normalization may be useful when dealing with sequences that are not all of the same length. For details on the applied normalization, see seqdist.

Value

A matrix of pairwise distances between sequences is returned.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

References

Pollock, Gary (2007) Holistic trajectories: a study of combined employment, housing and family careers by using multiple-sequence analysis. *Journal of the Royal Statistical Society: Series A* **170**, Part 1, 167–183.

See Also

seqsubm, seqdef, seqdist.

Examples

```
data(biofam)

## Building one channel per type of event left, children or married
bf <- as.matrix(biofam[, 10:25])
children <-  bf==4 | bf==5 | bf==6
married <- bf == 2 | bf== 3 | bf==6
left <- bf==1 | bf==3 | bf==5 | bf==6

## Building sequence objects
child.seq <- seqdef(children)
marr.seq <- seqdef(married)
left.seq <- seqdef(left)

## Using transition rates to compute substitution costs on each channel
mcdist <- seqdistmc(channels=list(child.seq, marr.seq, left.seq),
  method="OM", sm =list("TRATE", "TRATE", "TRATE"))

## Using a weight of 2 for children channel and specifying substitution-cost
smatrix <- list()
smatrix[[1]] <- seqsubm(child.seq, method="CONSTANT")
smatrix[[2]] <- seqsubm(marr.seq, method="CONSTANT")
smatrix[[3]] <- seqsubm(left.seq, method="TRATE")
mcdist2 <- seqdistmc(channels=list(child.seq, marr.seq, left.seq),
method="OM", sm =smatrix, cweight=c(2,1,1))
```

---

seqdss   *Extract distinct states sequence from a sequence object*

Description

Extract distinct states sequence from a sequence object.

Usage

```
seqdss(seqdata, with.missing=FALSE)
```

Arguments

seqdata	a sequence object as defined by the [seqdef](#) function.
with.missing	if set to TRUE, missing statuses (gaps in sequences) also appear in the DSS. See [seqdef](#) on options for handling missing values when creating sequence objects.

Details

Returns a sequence object containing the distinct states sequences, ie the durations are not taken into account. The DSS contained in 'D-D-D-D-A-A-A-A-A-A-A-D' is 'D-A-D'. Associated durations can be extracted with the [seqdur](#) function.

If called with the {with.missing=TRUE} argument, a missing state in a sequence is considered as the occurrence of an additional symbol of the alphabet, and two or more consecutive missing states are considered as two or more occurrences of the same state. Hence the DSS of A-A-*-*-*-B-B-C-C-D is A-*-B-C-D.

Value

a sequence object containing the distinct state sequence (DSS) for each sequence in the object given as argument.

Author(s)

Alexis Gabadinho

See Also

[seqdur](#).

Examples

```
## Creating a sequence object with the columns 13 to 24
## in the 'actcal' example data set
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## Retrieving the DSS
```

```
actcal.dss <- seqdss(actcal.seq)

## Displaying the DSS for the first 10 sequences
actcal.dss[1:10,]

## Example with with.missing argument
data(ex1)
ex1.seq <- seqdef(ex1, 1:13)

seqdss(ex1.seq)
seqdss(ex1.seq, with.missing=TRUE)
```

---

seqdur   *Extract state durations from a sequence object.*

Description

Extracts states durations from a sequence object. Returns a matrix containing the states durations for the sequences. The states durations in `'D-D-D-D-A-A-A-A-A-A-A-D'` are 4,7,1. Distinct states can be extracted with the [seqdss](#) function.

Usage

```
seqdur(seqdata, with.missing=FALSE)
```

Arguments

seqdata   a sequence object as defined by the [seqdef](#) function.

with.missing   if set to TRUE, durations are also computed for missing statuses (gaps in se-
               quences). See [seqdef](#) on options for handling missing values when creating
               sequence objects.

Value

a matrix containing the states durations for each distinct state in each sequence.

Author(s)

Alexis Gabadinho

See Also

[seqdss](#).

Examples

```
## Creating a sequence object with the columns 13 to 24
## in the 'actcal' example data set
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## Retrieving the DSS
actcal.dur <- seqdur(actcal.seq)

## Displaying the durations for the first 10 sequences
actcal.dur[1:10,]
```

---

seqeapplysub   *Checking for the presence of given event subsequences*

Description

Checks occurrences of the subsequences subseq among the event sequences and returns the result according to the selected method.

Usage

```
seqeapplysub(subseq, method = NULL, constraint = NULL,
           rules=FALSE)
```

Arguments

subseq	list of subsequences (an event subsequence object) such as created by seqefsub
method	type of result, should be one of "count", "presence" or "age"
constraint	Time constraints overriding those used to compute subseq. See seqeconstraint
rules	If set to TRUE, instead of checking occurrences of the subsequences among the event sequences, check the occurrence of the subsequences inside the subsequences (internally used by seqerules)

Details

There are three methods implemented: "count" counts the number of occurrence of each given subsequence in each event sequence; "presence" returns 1 if the subsequence is present, 0 otherwise; "age" returns the age of appearance of each subsequence in each event sequence. In case of multiple possibilities, the age of the first occurrence is returned. When the subsequence is not in the sequence, -1 is returned.

Value

The return value is a matrix where each row corresponds to a sequence (row names are set accordingly) and each column corresponds to a subsequence (col names are set accordingly). The cells of the matrix contain the requested values (count, presence-absence indicator or age).

Author(s)

Matthias Studer and Reto Bürgin (alternative counting methods) (with Gilbert Ritschard for the help page)

References

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2009). Mining Sequence Data in R with the TraMineR package: A user's guide. Department of Econometrics and Laboratory of Demography, University of Geneva.

See Also

[seqecreate](#) for more information on event sequence object and *Gabadinho et al. (2009)* on how to use the event sequence analysis module.

Examples

```
## Loading data
data(actcal.tse)

## Creating the event sequence object
actcal.seqe <- seqecreate(actcal.tse)

## Printing sequences
actcal.seqe[1:10]

## Looking for frequent subsequences
fsubseq <- seqefsub(actcal.seqe,pMinSupport=0.01)

## Counting the number of occurrences of each subsequence
msubcount <- seqeapplysub(fsubseq,method="count")
## First lines...
msubcount[1:10,1:10]
## Presence-absence of each subsequence
msubpres <- seqeapplysub(fsubseq,method="presence")
## First lines...
msubpres[1:10,1:10]

## Age at first appearance of each subsequence
msubage <- seqeapplysub(fsubseq,method="age")

## First lines...
msubage[1:10,1:10]
```

---

seqecmpgroup   *Identifying discriminating subsequences*

Description

Identify and sort the most discriminating subsequences by their discriminating power.

Usage

```
seqecmpgroup(subseq, group, method="chisq", pvalue.limit=NULL,
            weighted = TRUE)
```

Arguments

subseq	A subseqelist object (list of subsequences) such as produced by [seqefsub](#)
group	Group membership, i.e., a variable or factor defining the groups which we want to discriminate
method	The discrimination method; one of "bonferroni" or "chisq"
pvalue.limit	Can be used to filter the results. Only subsequences with a p-value lower than this parameter are selected. If NULL all subsequences are returned (regardless of their p-values).
weighted	Logical. If TRUE, seqecmpgroup uses the weights specified in subseq, (see [seqefsub](#)).

Details

The following discrimination test functions are implemented: chisq, the Pearson Independence Chi-squared test, and bonferroni, the Pearson Independence Chi-squared test with Bonferroni correction.

Value

An objet of type subseqelistchisq (subtype of subseqelist) with the following elements

subseq	Sorted list of found discriminating subsequences
seqe	The event sequence object on which the tests were computed
constraint	Time constraints used for searching the subsequences (see [seqeconstraint](#))
labels	Levels (value labels) of the target group variable
type	Type of test used
data	A data frame with columns support, index (original order of the subsequence) and a pair of frequency and Pearson residual columns for each group

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

References

Studer, M., Müller, N.S., Ritschard, G. & Gabadinho, A. (2010), "Classer, discriminer et visualiser des séquences d'événements", In Extraction et gestion des connaissances (EGC 2010), *Revue des nouvelles technologies de l'information* RNTI. Vol. E-19, pp. 37-48.

See Also

See also [plot.subseqelistchisq](#) to plot the results

Examples

```
data(actcal.tse)
actcal.seqe <- seqecreate(actcal.tse)

##Searching for frequent subsequences, that is, appearing at least 20 times
fsubseq <- seqefsub(actcal.seqe, pMinSupport=0.01)

##searching for susbsequences discriminating the most men and women
data(actcal)
discr <- seqecmpgroup(fsubseq, group=actcal$sex, method="bonferroni")
##Printing discriminating subsequences
print(discr)
##Plotting the six most discriminating subsequences
plot(discr[1:6])
```

---

seqeconstraint   *Setting time constraints and the counting method*

Description

Function used to set time constraints and the counting method in methods (seqe..) for event
sequences such as [seqefsub](#) for searching frequent subsequences or [seqeapplysub](#) for checking
occurrences of subsequences.

Usage

```
seqeconstraint(maxGap = -1, windowSize = -1, ageMin = -1,
    ageMax = -1, ageMaxEnd = -1, countMethod = 1)
```

Arguments

maxGap	The maximum time gap between two events
windowSize	The maximum time span accepted for subsequences
ageMin	Minimal start time position allowed for subsequences. Ignored when equal to -1 (default).
ageMax	Maximal start time position allowed for subsequences. Ignored when equal to -1 (default).
ageMaxEnd	Maximal end time position allowed for subsequences. Ignored when equal to -1 (default).
countMethod	By default, subsequences are counted only one time by sequence ('COBJ' method). Alternative counting methods are 'CDIST_O', 'CWIN', 'CMINWIN' or 'CDIST' respectively. See details.

Details

maxGap, windowSize, ageMin, ageMax and ageMaxEnd. If so, two events should not be separated by more than maxGap and the whole subsequence should not exceed a windowSize time span. The other parameters specify the start and end age of the subsequence, it should start between ageMin and ageMax and finish before ageMaxEnd. Parameters ageMin, ageMax and ageMaxEnd are interpreted as the number of positions (time units) from the beginning of the sequence.

There are 5 options for the countMethod argument. (1) By default, the count is the number of sequences that contain the subsequence ("COBJ" method). Alternatives are (2) "CDIST_O" (counts all distinct occurrences in each sequence including possibly overlapping occurrences, i.e., occurrences sharing a same event occurrence), (3) "CWIN" (number of slidden windows of length windowSize that contain an occurrence of the subsequence), (4) "CMINWIN" (number of minimal windows of occurrence) and (5) "CDIST" (distinct occurrences without event occurrences overlap). See references.

Value

A constraint object containing one item per constraint type.

Author(s)

Matthias Studer, Nicolas S. Müller and Reto Bürgin (alternative counting methods) (with Gilbert Ritschard for the help page)

References

Joshi, Mahesh V., George Karypis, and Vipin Kumar (2001) A Universal Formulation of Sequential Patterns *Proceedings of the KDD'2001 Workshop on Temporal Data Mining*, San Francisco.

Ritschard, G., A. Gabadinho, N.S. Müller and M. Studer (2008), Mining event sequences: A social science perspective, *International Journal of Data Mining, Modelling and Management, IJDMMM*, 1(1), 68-90.

See Also

[seqefsub](), [seqeapplysub]()

| seqecontain | *Check if sequence contains events* |

Description

Check if an event sequence or subsequence contains given events

Usage

```
seqecontain(seq, eventList, exclude = FALSE)
```

Arguments

seq	A event sequence object (seqelist) or a an event subsequence object (subseqelist)
eventList	A list of events
exclude	if TRUE the search is exclusive and returns FALSE for any subsequence containing an event that is not in eventList

Details

Checks, for each provided event sequence, if it contains one of the events in eventList. If exclude is TRUE, seqecontain looks if all events of the subsequence are in eventList.

Value

A logical vector.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

[seqecreate](#) for creating event sequence objects and [seqefsub](#) for creating event subsequence objects.

Examples

```
data(actcal.tse)
actcal.seqe <- seqecreate(actcal.tse)

##Searching for frequent subsequences, that is appearing at least 20 times
fsubseq <- seqefsub(actcal.seqe,minSupport=20)

##looking for subsequence with FullTime
seqecontain(fsubseq,c("FullTime"))
```

---

seqecreate   *Create event sequence objects.*

Description

Create an event sequence object either from time stamped events or from a state sequence object.

Usage

```
seqecreate(data = NULL, id = NULL, timestamp = NULL, event = NULL,
    endEvent = NULL, tevent =  "transition", use.labels=TRUE,
    weighted=TRUE)
```

Arguments

data	A state sequence object (see [seqdef](#)) or a data frame
id	The sequence 'id' (integer) column when data are provided in TSE format (ignored if `data` argument is provided).
timestamp	The event 'timestamp' (double) column when data are provided in TSE format, i.e., the time at which events occur (ignored if `data` argument is provided).
event	The 'event' column when data are provided in TSE format, i.e., the events occurring at the specified time stamps (ignored if `data` argument is provided).
endEvent	If specified this event serves as a flag for the end of observation time (total length of event sequences).
tevent	Either a transition matrix or a method to generate events from state sequences (see [seqetm](#)). Used only when `data` is a state sequence object.
use.labels	If `TRUE`, transitions names are built from long state labels rather than from the short state names of the alphabet.
weighted	If `TRUE` and `data` is a state sequence object, use the weights specified in `data` (see [seqdef](#))

Details

There are several ways to create an event sequence object. The first one is by providing the events in TSE format (see [seqformat](#)), i.e. by providing three paired lists: id, timestamp and event, such that each triplet (id, timestamp, event) defines the event that occurs at time timestamp for case id. Several events at the same time for a same id are allowed. The lists can be provided with the arguments id, timestamp and event. An alternative is by providing a data frame as `data` argument in which case the function takes the required information from the "id", "timestamp" and "event" columns of that data frame. In any case with TSE format, **listed events should be grouped by id** and an error will be thrown otherwise. Such grouping can be achieved by ordering the data according to the id column using the [order](#) function (e.g., data[order(data$id),]).

The other way is to pass a state sequence object (as `data` argument) and to perform an automatic state-to-event conversion. The simplest way to make a conversion is by means of a predefined method (see [seqetm](#)), such as "transition" (one distinct event per possible transition), "state" (a new event for each entering in a new state) and "period" (a pair of events, one start-state event and one end-state event for each found transition). For a more customized conversion, you can specify a transition matrix in the same way as in [seqformat](#). Function [seqetm](#) can help you in creating your transition matrix.

Event sequence objects as created by seqecreate are required by most other 'seqe' methods, such as [seqefsub](#) or [seqeapplysub](#) for example.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

[seqformat](#) for converting between sequence formats, [seqefsub](#) for searching frequent subsequences, [seqecmpgroup](#) to search for discriminant subsequences, [seqeapplysub](#) for counting subsequence

occurrences, `seqelength` for information about length (observation time) of event sequences, `seqdef`
to create a state sequence object.

Examples

```
##Starting with states sequences
##Loading data
data(biofam)
## Creating state sequences
biofam.seq <- seqdef(biofam,10:25,informat='STS')
## Creating event sequences from biofam
biofam.seqe <- seqecreate(biofam.seq)

## Loading data
data(actcal.tse)
## Creating sequences
actcal.seqe <- seqecreate(id=actcal.tse$id, timestamp=actcal.tse$time,
event=actcal.tse$event)
##printing sequences
actcal.seqe[1:10]
## Using the data argument
actcal.seqe <- seqecreate(data=actcal.tse)
```

---

seqefsub   *Searching for frequent subsequences*

Description

Returns the list of subsequences with minimal support sorted in decreasing order of support. Various
time constraints can be set to restrict the search to specific time periods or subsequence durations.
The function permits also to get information on specified subsequences.

Usage

```
seqefsub(seq, strsubseq = NULL, minSupport = NULL,
    pMinSupport = NULL, constraint = seqeconstraint(), maxK = -1,
    weighted = TRUE)
```

Arguments

seq	A list of event sequences
strsubseq	A list of specific subsequences to look for. See details.
minSupport	The minimum support (in number of sequences)
pMinSupport	The minimum support (in percentage, will be rounded)
constraint	A time constraint object as returned by `seqeconstraint`
maxK	The maximum number of events allowed in a subsequence
weighted	Logical. If TRUE, seqefsub use the weights specified in seq (see `seqeweight`).

Details

There are two usages of this function. The first is for searching subsequences satisfying a support condition. By default, the support is counted per sequence and not per occurrence, i.e. when a sequence contains twice a same subsequence it is counted only once. Use the countMethod argument of seqeconstraint to change that. The minimal required support can be set with pMinSupport as a proportion (between 0 and 1) in which case it will be rounded, or through minSupport as a number of sequences. Time constraints can also be imposed with the constraint argument, which must be the outcome of a call to the seqeconstraint function).

The second possibility is for searching sequences that contain specified subsequences. This is done by passing the list of subsequences with the strsubseq argument. The subsequences must be in the same format as that used to display subsequences (see str.seqelist). Each transition (group of events) should be enclosed in parentheses () and separated with commas, and the succession of transitions should be denoted by a '-' indicating a time gap. For instance "(FullTime)-(PartTime, Children)" stands for the subsequence "FullTime" followed by the transition defined by the two simultaneously occurring events "PartTime" and "Children".

Information about the sequences that contain the subsequences can then be obtained with the seqeapplysub function.

Subsets of the returned subseqelist can be accessed with the [] operator (see example). There are print and plot methods for subsequelist.

Value

A subseqelist object which contain at least the following objects:

seqe	The list of sequences in which the subsequences were searched (a seqelist event sequence object).
subseq	A list of subsequences (a seqelist event sequence object).
data	A data frame containing details (support, frequency, ...) about the subsequences
constraint	The constraint object used when searching the subsequences.
type	The type of search: 'frequent' or 'user'

Author(s)

Matthias Studer and Reto Bürgin (alternative counting methods) (with Gilbert Ritschard for the help page)

See Also

See plot.subseqelist to plot the result. See seqecreate for creating event sequences. See seqeapplysub to count the number of occurrences of frequent subsequences in each sequence. See is.seqelist about seqelist.

Examples

```
data(actcal.tse)
actcal.seqe <- seqecreate(actcal.tse)
```

```
##Searching for frequent subsequences, that is, appearing at least 20 times
fsubseq <- seqefsub(actcal.seqe, minSupport=20)
##The same using a percentage
fsubseq <- seqefsub(actcal.seqe, pMinSupport=0.01)
##Getting a string representation of subsequences
##Ten first subsequences
fsubseq[1:10]

##Using time constraints
##Looking for subsequence starting in summer (between june and september)
fsubseq <- seqefsub(actcal.seqe, minSupport=10,
  constraint=seqeconstraint(ageMin=6, ageMax=9))
fsubseq[1:10]

##Looking for subsequence contained in summer (between june and september)
fsubseq <- seqefsub(actcal.seqe, minSupport=10,
  constraint=seqeconstraint(ageMin=6, ageMax=9, ageMaxEnd=9))
fsubseq[1:10]

##Looking for subsequence enclosed in a 6 month period
## and with a maximum gap of 2 month
fsubseq <- seqefsub(actcal.seqe, minSupport=10,
  constraint=seqeconstraint(maxGap=2, windowSize=6))
fsubseq[1:10]
```

---

seqeid   *Retrieve unique ids from an event sequence object.*

Description

Retrieve the unique ids from an event sequence object or from a list of event sequence object.

Usage

```
seqeid(s)
```

Arguments

s   An event sequence object (as created with [seqecreate](#)) or a list of event se-
                quence objects

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

Examples

```
data(actcal.tse)
actcal.seqe <- seqecreate(actcal.tse)
seqeid(actcal.seqe)
```

---

seqelength   *Lengths of event sequences*

Description

The length of an event sequence is its time span, i.e., the total time of observation. This information is useful to perform for instance a survival analysis. The function seqelength retrieves the lengths of the provided sequences, while seqelength <- sets the length of the sequences.

seqesetlength is deprecated.

Usage

```
seqelength(s)
seqelength(s) <- value
seqesetlength(s, len)
```

Arguments

s   An event sequence object (seqelist).

len   A list of sequence lengths.

value   A list of sequence lengths.

Value

A numeric vector with the lengths of the sequences.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

Examples

```
data(actcal.tse)
actcal.seqe <- seqecreate(actcal.tse)
## Since endEvent is not specified, contains no sequence lengths
## We set them manually as 12 for all sequences
sl <- numeric()
sl[1:2000] <- 12
seqelength(actcal.seqe) <- sl
actcal.seqe[1:10]
## Retrieve lengths
seqelength(actcal.seqe)
```

---

seqetm   *Create a transition-definition matrix*

Description

This function automatically creates a transition-definition matrix from a state sequence object to transform the state sequences into time stamped event sequences (in TSE format).

Usage

```
seqetm(seq, method = "transition", use.labels = TRUE,
  sep = ">", bp = "", ep = "end")
```

Arguments

seq	State sequence object from which transition events will be determined
method	The method to use. One of "transition", "period" or "state".
use.labels	If TRUE, transition names are built from state labels rather than from the alphabet.
sep	Separator to be used between the from-state and to-state that define the transition ("transition" method).
bp	Prefix for beginning of period event names ("period" method)
ep	Prefix for end of period event names ("period" method)

Details

Warning!!!: State labels should not contain commas ", " which are reserved for separating multiple events of a same transition!

One of three methods can be selected with the method argument:

"transition" generates a single (from-state > to-state) event for each found transition and a distinct start-state event for each different sequence start;

"period" generates a pair of events (end-state-event, start-state-event) for each found transition, a start-state event for the beginning of the sequence and an end-state event for the end of the sequence; names used for end-state and start-state names can be controlled with the bp and ep arguments;

"state" generates only the to-state event of each found transition (useful for analysing state sequences with methods for event sequences);

Value

The transition-definition matrix.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

seqformat for converting to TSE format, seqecreate for creating an event sequence object, seqdef for creating a state sequence object.

Examples

```
## Creating a state sequence object from columns 13 to 24
## in the 'actcal' example data set
data(actcal)
actcal.seq <- seqdef(actcal,13:24,
        labels=c("FullTime", "PartTime", "LowPartTime", "NoWork"))
## Creating a transition matrix, one event per transition
seqetm(actcal.seq,method = "transition")

## Creating a transition matrix, single to-state events
seqetm(actcal.seq,method = "state")

## Creating a transition matrix, two events per transition
seqetm(actcal.seq,method = "period")

## changing the prefix of period start event.
seqetm(actcal.seq,method = "period", bp="begin")
```

---

seqeweight   *Setting or retrieving weights of an event sequence object.*

Description

Event sequence objects can be weighted. Weights are used by other functions such as seqefsub or seqecmpgroup to compute weighted statistics.

Usage

```
seqeweight(s)
seqeweight(s) <- value
```

Arguments

s   An event sequence object (seqelist).

value   Numerical vector containing weights

Value

seqeweight returns a numerical vector containing the weights associated to each event sequence.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

Examples

```
##Starting with states sequences
##Loading data
data(biofam)
## Creating state sequences
biofam.seq <- seqdef(biofam,10:25,informat='STS')

## Creating event sequences from biofam
biofam.seqe <- seqecreate(biofam.seq, weighted=FALSE)

## Using the weights
seqeweight(biofam.seqe) <- biofam$wp00tbgs

## Now seqefsub accoounts for weights unless weighted is set to FALSE
fsubseq <- seqefsub(biofam.seqe, pMinSupport=0.01)

## Searching for weighted susbsequences which best
## discriminate the birth cohort
discr <- seqecmpgroup(fsubseq, group=biofam$birthyr>=1940)
plot(discr[1:15])
```

---

seqfind   *Indexes of state sequence(s) x in state sequence object y*

Description

Finds the row indexes of state sequence(s) x in the state sequence object y.

Usage

```
seqfind(x, y)
```

Arguments

x   a state sequence object containing one or more sequences ([seqdef](#)).

y   a state sequence object.

Value

row index(es) of sequence(s) x in the set of sequences y.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

See Also

.

Examples

```
data(mvad)
mvad.shortlab <- c("EM", "FE", "HE", "JL", "SC", "TR")
mvad.seq <- seqdef(mvad, states=mvad.shortlab, 15:86)

## Finding occurrences of sequence 176 in mvad.seq
seqfind(mvad.seq[176,],mvad.seq)

## Finding occurrences of sequence 1 to 8 in mvad.seq
seqfind(mvad.seq[1:8,],mvad.seq)
```

---

seqformat   *Conversion between sequence formats*

Description

Convert a sequence data set from one format to another.

Usage

```
seqformat(data, var=NULL, id=NULL,
         from, to, compressed=FALSE,
         nrep=NULL, tevent, stsep=NULL, covar=NULL,
         SPS.in=list(xfix="()", sdsep=","),
         SPS.out=list(xfix="()", sdsep=","),
         begin=NULL, end=NULL, status=NULL,
         process=TRUE, pdata=NULL, pvar=NULL,
         limit=100, overwrite=TRUE,
         fillblanks=NULL, tmin=NULL, tmax=NULL, nr="*")
```

Arguments

data	a data frame or matrix containing sequence data.
var	List of columns with the sequence data. Default is NULL, i.e., all columns. Sequences are assumed to be in compressed form (character strings) when there is a single column and in extended form otherwise.
id	Column containing the 'id' of the sequences. Mandatory with from="SPELL" in order to identify the spells of a same sequence.
from	Format of the input data. One of "STS", "SPS", "SPELL". If data is a sequence object, format is automatically set to "STS".
to	Format for output data. One of "STS", "SPS", "SRS", "DSS", "TSE".
compressed	Logical. Should "STS", "SPS" or "DSS" output be compressed into character strings? Ignored for other output formats.
nrep	Number of shifted replications for output in "SRS" format.

tevent	Transition definition matrix for converting to time-stamped-event ("TSE") format. Should be a matrix of size $d * d$ where d is the number of distinct states appearing in the sequences. In this matrix, the cell (i, j) lists the events associated with a transition from state i to state j.
stsep	Separator character between successive elements in compressed (character strings) input data. If NULL (default value), the seqfcheck function is called for detecting automatically a separator among "-" and ":". Other separators must be specified explicitly.
covar	When from="STS" or from="SPS", additional column names to be included as covariates in the output data frame. When to="SRS" the covariates are replicated across the shifted replicated rows. Default is NULL. Ignored when from="SPELL".
SPS.in	List with the xfix= and sdsep= specifications for the state-duration couples in input data in SPS form. The first specification, xfix, specifies the prefix/suffix character (use a two-character string if the prefix and suffix differ and set xfix="" when no prefix/suffix are present). The second one, sdsep, specifies the state/duration separator.
SPS.out	List with the xfix and sdsep specifications for output in SPS format. (see argument SPS.in above.)
nr	Symbol used for missing state in input "SPS" format which will be converted to NA in "STS" representation.
begin	When converting from SPELL, the column with the beginning position of the spell
end	When converting from SPELL, the column with the end position of the spell
status	When converting from SPELL, the column with the status
process	Logical: When converting from SPELL, should sequences be created on a process time axis? Default is TRUE. Set as FALSE for creating sequences on a calendar time axis.
pdata	When converting from SPELL and process=TRUE, either NULL, "auto" or the name of the data frame containing the individual 'birth' time, that is, the initial time from which the process time will be computed. If set as NULL (default), the starting and ending time of each spell are supposed to be ages. If set as "auto", ages are computed using the starting time of the first spell of each individual as her/his birth date. If external birth dates are provided, the pdata data must contain two columns: an id to match the birth time with SPELL data and a 'birth' time.
pvar	When pdata is a data frame, a vector of two names or numbers, the first one specifying the column with the individual 'id', and the second one the 'birth' time.
limit	When converting from SPELL, size of the resulting data frame when creating age sequences (by default ranges from age 1 to age 100)
overwrite	When converting from SPELL, if overwrite is set to TRUE, the most recent episode overwrites the older one when they overlap each other. If set to FALSE, the most recent episode starts in case of overlap after the end of the previous one.

fillblanks	When converting from SPELL, if `fillblanks` is not `NULL`, gaps between episodes are filled with the `fillblanks` character value.
tmin	Integer. When converting from SPELL with `process=FALSE`, defines the starting time of the axis. If set as `NULL`, the minimum time is taken from the 'begin' column in the data.
tmax	Integer. When converting from SPELL with `process=FALSE`, defines the ending time. If set as `NULL`, the value is guessed from the data (not so accurately!).

Details

The `seqformat` function is used to convert data from one format to another. The input data is first converted into the STS format and then converted to the output format. Depending on input and output formats, some information can be lost in the conversion process. The output is a matrix, NOT a sequence object to be passed to TraMineR functions for plotting and mining sequences (use the [seqdef](#) function for that). See *Gabadinho et al. (2009)* and *Ritschard et al. (2009)* for more details on longitudinal data formats and converting between them.

Value

A data frame

Author(s)

Alexis Gabadinho, Nicolas S. Müller and Matthias Studer (with Gilbert Ritschard for the help page)

References

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2009). Mining Sequence Data in R with the TraMineR package: A user's guide. Department of Econometrics and Laboratory of Demography, University of Geneva.

Ritschard, G., A. Gabadinho, M. Studer and N. S. Müller. Converting between various sequence representations. in Ras, Z. & Dardzinska, A. (ed.) *Advances in Data Management*, Springer, 2009, 223, 155-175.

See Also

[seqdef](#)

Examples

```
## Converting sequences into SPS format
data(actcal)
actcal.SPS.A <- seqformat(actcal,13:24, from="STS", to="SPS")
head(actcal.SPS.A)

## SPS (compressed) format with no prefix/suffix "/" as state/duration separator
actcal.SPS.B <- seqformat(actcal,13:24,
from="STS", to="SPS", compressed=TRUE,
SPS.out=list(xfix="", sdsep="/"))
head(actcal.SPS.B)
```

```
## Converting sequences into DSS (compressed) format
actcal.DSS <- seqformat(actcal,13:24,
from="STS", to="DSS", compressed=TRUE)
head(actcal.DSS)
```

---

seqfpos   *Search for the first occurrence of a given element in a sequence*

Description

Returns a vector containing the position of the first occurrence of the given element in each of the
sequences in the data set.

Usage

```
seqfpos(seqdata, state)
```

Arguments

seqdata	a sequence object (see [seqdef](#) function).
state	the state element to search in the sequences

Details

the state to search for has to be passed as a character string, and must be one of the state returned by
the [alphabet](#) function. If the state is not contained in a sequence, NA is returned for this sequence.

Author(s)

Alexis Gabadinho

Examples

```
data(biofam)
biofam.seq <- seqdef(biofam,10:25)

## Searching for the first occurrence of state 1
## in the biofam data set.
seqfpos(biofam.seq,"1")
```

---

seqgen   *Random sequences generation*

Description

Generates random sequences.

Usage

```
seqgen(n, length, alphabet, p)
```

Arguments

n	number of sequences to generate
length	sequences length
alphabet	the alphabet from which the sequences are generated
p	an optional vector of probabilities for the states in the alphabet. Must be of the same length as the alphabet. If not specified, equal probabilities are used.

Details

Each sequence is generated by choosing a set of random numbers (with min=1 and max=length of the alphabet) using the [runif](#) function. When the probability distribution is not specified, the uniform probability distribution giving same probability to each state is used to generate the sequences.

Value

a sequence object.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

Examples

```
seq <- seqgen(1000,10,1:4,c(0.2,0.1,0.3,0.4))
seqstatd(seqdef(seq))
```

seqici	*Complexity index of individual sequences*

Description

Computes the complexity index, a composite measure of sequence complexity. The index uses the number of transitions in the sequence as a measure of the complexity induced by the state ordering and the longitudinal entropy as a measure of the complexity induced by the state distribution in the sequence.

Usage

```
seqici(seqdata, with.missing=FALSE)
```

Arguments

seqdata	a sequence object as returned by the the seqdef function.
with.missing	if set to TRUE, missing status (gaps in sequences) is handled as an additional state when computing the state distribution and the number of transitions in the sequence.

Details

The *complexity index* $C(s)$ of a sequence s is

$$C(s) = \sqrt{\frac{q(s)}{q_{max}} \frac{h(s)}{h_{max}}}$$

where $q(s)$ is the number of transitions in the sequence, q_{max} the maximum number of transitions, $h(s)$ the within entropy, and h_{max} the theoretical maximum entropy which is $h_{max} = -\log 1/|A|$.

The index $C(s)$ is the geometric mean of its two components which are normalized. The minimum value of 0 can only be reached by a sequence made of one distinct state, containing thus 0 transitions and having an entropy of 0. The maximum 1 of $C(s)$ is reached when the two following conditions are fulfilled: i) Each of the state in the alphabet is present in the sequence and the total durations are uniform, that is, equal to ℓ/a and ii) The number of transitions in the sequence is equal to $\ell - 1$, that is, the length ℓ_d of the DSS is equal to the length of the sequence ℓ

Value

a vector of length equal to the number of sequences in seqdata containing the complexity index value of each sequence.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

Gabadinho, A., Ritschard, G., Studer, M. and Müller, N.S. (2010). "Indice de complexité pour le tri et la comparaison de séquences catégorielles", In *Extraction et gestion des connaissances (EGC 2010), Revue des nouvelles technologies de l'information RNTI.* Vol. E-19, pp. 61-66.

See Also

seqient, seqST

Examples

```
## Creating a sequence object from the mvad data set
data(mvad)
mvad.labels <- c("employment", "further education", "higher education",
                 "joblessness", "school", "training")
mvad.scodes <- c("EM","FE","HE","JL","SC","TR")
mvad.seq <- seqdef(mvad, 15:86, states=mvad.scodes, labels=mvad.labels)

##
mvad.ci <- seqici(mvad.seq)
summary(mvad.ci)
hist(mvad.ci)

## Example using with.missing argument
data(ex1)
ex1.seq <- seqdef(ex1, 1:13)
seqici(ex1.seq)
seqici(ex1.seq, with.missing=TRUE)
```

---

seqient | *Within sequence entropies*

Description

Computes normalized or non-normalized within sequence entropies

Usage

```
seqient(seqdata, norm=TRUE, base=exp(1), with.missing=FALSE)
```

Arguments

seqdata	a sequence object as returned by the the seqdef function.
norm	logical: should the entropy be normalized? TRUE by default. (see details)

base	real positive value: base of the logarithm used in the entropy formula (see details). If entropy is normalized (norm=TRUE), its value is the same whatever the base. Default is exp(1), i.e., the natural logarithm is used.
with.missing	logical: if TRUE, the missing state (gap in sequences) is handled as an additional state when computing the state distribution in the sequence.

Details

The seqient function returns the Shannon entropy of each sequence in seqdata. The entropy of a sequence is computed using the formula

$$h(\pi_1, \ldots, \pi_s) = -\sum_{i=1}^{s} \pi_i \log \pi_i$$

where s is the size of the alphabet and π_i the proportion of occurrences of the ith state in the considered sequence. The log is here the natural logarithm, i.e., the logarithm in base e. The entropy can be interpreted as the 'uncertainty' of predicting the states in a given sequence. If all states in the sequence are the same, the entropy is equal to 0. The maximum entropy for a sequence of length 12 with an alphabet of 4 states is 1.386294 and is attained when each of the four states appears 3 times.

Normalization can be requested with the norm=TRUE option, in which case the returned value is the entropy divided by the entropy of the alphabet. The later is an upper bound for the entropy of sequences made from this alphabet. It exactly is the maximal entropy when the sequence length is a multiple of the alphabet size. The value of the normalized entropy is independent of the chosen logarithm base.

Value

a vector with an entropy value for each sequence in seqdata; the vector length is equal to the number of sequences.

Author(s)

Alexis Gabadinho

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2009). Mining Sequence Data in R with the TraMineR package: A user's guide. *Department of Econometrics and Laboratory of Demography, University of Geneva*.

See Also

seqstatd for the entropy of the transversal state distributions by positions in the sequence.

Examples

```
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## Summarize and plots an histogram
## of the within sequence entropy
actcal.ient <- seqient(actcal.seq)
summary(actcal.ient)
hist(actcal.ient)

## Examples using with.missing argument
data(ex1)
ex1.seq <- seqdef(ex1, 1:13, weights=ex1$weights)

seqient(ex1.seq)
seqient(ex1.seq, with.missing=TRUE)
```

---

seqistatd   *State frequencies in each individual sequence*

Description

Returns the state frequencies (total durations) for each sequence in the sequence object.

Usage

```
 seqistatd(seqdata, with.missing=FALSE, prop=FALSE)
```

Arguments

seqdata	a sequence object (see [seqdef](#) function).
with.missing	logical: if set as TRUE, total durations are also computed for the missing status (gaps in the sequences). See [seqdef](#) on options for handling missing values when creating sequence objects.
prop	logical: if TRUE, proportions of time spent in each state are returned instead of absolute values. This option is specially useful when sequences contain missing states, since the sum of the state durations may not be the same for all sequences.

Author(s)

Alexis Gabadinho

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

Examples

```
data(actcal)
actcal.seq <- seqdef(actcal,13:24)
seqistatd(actcal.seq[1:10,])

## Example using "with.missing" argument
data(ex1)
ex1.seq <- seqdef(ex1, 1:13, weights=ex1$weights)

seqistatd(ex1.seq)
seqistatd(ex1.seq, with.missing=TRUE)
```

---

seqlegend   *Plot a legend for the states in a sequence object*

Description

Plots a legend for the states in a sequence object. Useful if several graphics are plotted together and only one legend is necessary. Unless specified by the user, the *cpal* and *labels* attributes of the sequence object are used for the colors and text appearing in the legend (see [seqdef](#)).

Usage

```
seqlegend(seqdata, with.missing="auto",
          cpal=NULL, missing.color=NULL, ltext=NULL,
          position="topleft", fontsize=1, ...)
```

Arguments

seqdata   a sequence object as returned by the the [seqdef](#) function.

with.missing   if set to "auto" (default), a legend for the missing state is added automatically
                 if one or more of the sequences in seqdata contains a missing state. If TRUE a
                 legend for the missing state is added in any case. Setting to FALSE omits the
                 legend for the missing state.

cpal   alternative color palette to use for the states. If user specified, a vector of colors
                 with number of elements equal to the number of distinct states. By default, the
                 'cpal' attribute of the 'seqdata' sequence object is used (see [seqdef](#)).

missing.color   alternative color for representing missing values inside the sequences. By de-
                 fault, this color is taken from the "missing.color" attribute of the sequence object
                 being plotted.

ltext   optional description of the states to appear in the legend. Must be a vector of
                 character strings with number of elements equal to the number of distinct states.
                 If unspecified, the 'labels' attributes of the 'seqdata' sequence object is used (see
                 [seqdef](#)).

position	the position of the legend in the graphic area. For accepted values, see [legend](). Defaults to "topleft".
fontsize	size of the font for the labels. A value less than 1 decreases the font size, a value greater than 1 increases the font size. Defaults to 1.
...	optional arguments passed to the [legend]() function.

Author(s)

Alexis Gabadinho

Examples

```
## Loading the 'actcal' example data set
## and defining a sequence object with
## (activity statuses from jan. to dec. 2000)
## the data in columns 13 to 24
data(actcal)
actcal.seq <- seqdef(actcal,13:24,
labels=c("> 37 hours", "19-36 hours", "1-18 hours", "no work"))

## Plotting the sequences frequency,
## the states distribution
## and the legend
par(mfrow=c(2,2))
seqiplot(actcal.seq, tlim=0, withlegend=FALSE, border=NA, space=0)
seqfplot(actcal.seq, pbarw=TRUE, withlegend=FALSE)
seqdplot(actcal.seq, withlegend=FALSE)
seqlegend(actcal.seq)
```

---

seqlength	*Sequence length*

Description

Returns the length of sequences.

Usage

```
seqlength(seqdata)
```

Arguments

seqdata	a sequence object created with the [seqdef]() function.

Details

The length of a sequence is computed by eliminating the missing values at the end (right) and counting the number of states or events. The seqlength function returns a vector containing the length of each sequence in the sequence object given as argument.

Author(s)

Alexis Gabadinho

Examples

```
## Loading the 'famform' example data set
data(famform)

## Defining a sequence object with the 'famform' data set
ff.seq <- seqdef(famform)

## Retrieving the length of the first 10 sequences
## in the ff.seq sequence object
seqlength(ff.seq)
```

---

seqLLCP   *Compute the length of the longest common prefix of two sequences*

Description

Returns the length of the longest common prefix of two sequences. This attribute is described in *Elzinga (2008)*.

Usage

```
seqLLCP(seq1, seq2)
```

Arguments

seq1   a sequence from a sequence object.

seq2   a sequence from a sequence object.

Value

an integer being the length of the longest common prefix of the two sequences.

References

Elzinga, Cees H. (2008). Sequence analysis: Metric representations of categorical time series. *Technical Report*, Department of Social Science Research Methods, Vrije Universiteit, Amsterdam.

See Also

seqdist

Examples

```
data(famform)
famform.seq <- seqdef(famform)

## The LCP's length between sequences 1 and 2
## in the famform sequence object is 2
seqLLCP(famform.seq[1,],famform.seq[2,])
```

---

seqLLCS   *Compute the length of the longest common subsequence of two se-*
                           quences

Description

Returns the length of the longest common subsequence of two sequences. This attribute is described
in *Elzinga (2008)*.

Usage

```
seqLLCS(seq1, seq2)
```

Arguments

seq1   a sequence from a sequence object

seq2   a sequence from a sequence object

Value

an integer being the length of the longest common subsequence of the two sequences.

References

Elzinga, Cees H. (2008). Sequence analysis: Metric representations of categorical time series.
Technical Report, Department of Social Science Research Methods, Vrije Universiteit, Amsterdam.

See Also

[seqdist](#)

Examples

```
LCS.ex <- c("S-U-S-M-S-U", "U-S-SC-MC", "S-U-M-S-SC-UC-MC")
LCS.ex <- seqdef(LCS.ex)
seqLLCS(LCS.ex[1,],LCS.ex[3,])
```

---

seqlogp   *Logarithm of the probabilities of state sequences*

Description

Compute the logarithm of the probability of each state sequence obtained from a state transition
model. The probability of a sequence is equal to the product of each state probability of the se-
quence. There are several methods to compute a state probability.

Usage

```
seqlogp(seqdata, prob="trate", time.varying=TRUE,
        begin="freq", weighted=TRUE)
```

Arguments

seqdata   The sequence to compute the probabilities.

prob   either the name ("trate" or "freq"$ of the probability model to use to compute
                  the state probabilities, or an array specifying the transition probabilities at each
                  position t (see details).

time.varying   Logical. If TRUE, the probabilities (transitions or frequencies) are computed
                  separately for each time t point.

begin   Model used to compute the probability of the first state. Either "freq" to use the
                  observed frequencies on the first period or a vector specifying the probability of
                  each state of the alphabet.

weighted   Logical. If TRUE, uses the weights specified in seqdata when computing the
                  observed transition rates.

Details

The sequence likelihood $P(s)$ is defined as the product of the probability with which each of its
observed successive state is supposed to occur at its position. Let $s = s_1 s_2 \cdots s_\ell$ be a sequence of
length ℓ. Then
$$P(s) = P(s_1, 1) \cdot P(s_2, 2) \cdots P(s_\ell, \ell)$$
with $P(s_t, t)$ the probability to observe state s_t at position t.

The question is how to determinate the state probabilities $P(s_t, t)$. Several methods are available
and can be set using the prob argument.

One commonly used method for computing them is to postulate a Markov model, which can be
of various order. We can consider probabilities derived from the first order Markov model, that is,
each $P(s_t, t)$, $t > 1$ is set as the transition rate $p(s_t | s_{t-1})$. This is available in seqlogp by setting
prob="trate".
The transition rates may be considered constant over time/positions (time.varying=FALSE), that
is estimated across sequences from the observations at positions t and $t - 1$ for all t together.
Time varying transition rates may also be considered (time.varying=TRUE), in which case they
are computed separately for each position, that is estimated across sequences from the observations

at positions t and $t-1$ for each t, yielding an array of transition matrices. The user may also specify his own transition rates array or matrix.

Another method is to use the frequency of a state at each position to set $P(s_t, t)$ (prob="freq"). In the latter case, the probability of a sequence is independent of the probability of the transitions. Here again, the frequencies can be computed all together (time.varying=FALSE) or separately for each position t (time.varying=TRUE).

For $t = 1$, we set $P(s_1, 1)$ to the observed frequency of the state s_1 at position 1. Alternatively, the begin argument allows to specify the probability of the first state.

The likelihood $P(s)$ being generally very small, seqlogp return $-\log P(s)$. The latter quantity is minimal when $P(s)$ is equal to 1.

Value

A vector containing the logarithm of each sequence probability.

Author(s)

Matthias Studer and Alexis Gabadinho (with Gilbert Ritschard for the help page)

Examples

```
## Creating the sequence objects using weigths
data(biofam)
biofam.seq <-  seqdef(biofam, 10:25, weights=biofam$wp00tbgs)

## Computing sequence probabilities
biofam.prob <- seqlogp(biofam.seq)
## Comparing the probability of each cohort
cohort <- biofam$birthyr>1940
boxplot(biofam.prob~cohort)
```

---

seqmeant   *Mean durations in each state*

Description

Compute the mean total time spent in each state of the alphabet for the set of sequences given as input.

Usage

```
seqmeant(seqdata, weighted=TRUE, with.missing=FALSE, prop=FALSE, serr=FALSE)
```

Arguments

seqdata	a sequence object as defined by the [seqdef](#) function.
weighted	logical: if TRUE, the weights (weights attribute) attached to the sequence object are used for computing weighted mean total time.
with.missing	logical: if set to TRUE, cumulated durations are also computed for the missing status (gaps in the sequences). See [seqdef](#) on options for handling missing values when creating sequence objects.
prop	logical: if TRUE, proportions of time spent in each state are returned instead of absolute values. This option is especially useful when sequences contain missing states, since the sum of the state durations may not be the same for all sequences.
serr	logical: if TRUE, the variance and standard deviation of the total time spent in the states, as well as the standard error of the mean are also computed.

Value

An object of class *stslist.meant*. There are `print` and `plot` methods for such objects.

Author(s)

Alexis Gabadinho

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

See Also

[plot.stslist.meant](#) for basic plots of *stslist.meant* objects and [seqmtplot](#) ([seqplot](#) with `type="mt"`) argument for more sophisticated plots of the mean durations allowing grouping and legend.

Examples

```
## Defining a sequence object with columns 13 to 24
## in the actcal example data set
data(actcal)
actcal.lab <- c("> 37 hours", "19-36 hours", "1-18 hours", "no work")
actcal.seq <- seqdef(actcal,13:24,labels=actcal.lab)

## Computing the mean time in the different states
seqmeant(actcal.seq)

## Mean times with their standard error
seqmeant(actcal.seq, serr=TRUE)
```

---

seqmodst   *Sequence of modal states*

Description

Sequence made of the modal state at each position.

Usage

```
seqmodst(seqdata, weighted=TRUE, with.missing=FALSE)
```

Arguments

seqdata	a state sequence object as defined by the [seqdef](#) function.
weighted	if TRUE, distributions account for the weights assigned to the state sequence object (see [seqdef](#)). Set as FALSE if you want ignore the weights.
with.missing	If FALSE (default value), returned distributions ignore missing values.

Details

In case of multiple modal states at a given position, the first one is taken. Hence, the result may vary with the alphabet order.

Value

an object of class *stslist.modst*. This is actually a state sequence object (containing a single state sequence) with additional attributes, among which the Frequencies attribute containing the transversal frequency of each state in the sequence. There are print and plot methods for such objects. More sophisticated plots can be produced with the seqplot function.

Author(s)

Alexis Gabadinho

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

See Also

[plot.stslist.modst](#) for default plot method, [seqplot](#) for higher level plots.

Examples

```
## Defining a sequence object with the data in columns 10 to 25
## (family status from age 15 to 30) in the biofam data set
data(biofam)
biofam.lab <- c("Parent", "Left", "Married", "Left+Marr",
"Child", "Left+Child", "Left+Marr+Child", "Divorced")
biofam.seq <- seqdef(biofam, 10:25, labels=biofam.lab)

## Modal state sequence
seqmodst(biofam.seq)

## Examples using weights and with.missing arguments
data(ex1)
ex1.seq <- seqdef(ex1, 1:13, weights=ex1$weights)

seqmodst(ex1.seq)
seqmodst(ex1.seq, weighted=FALSE)
seqmodst(ex1.seq, weighted=FALSE, with.missing=TRUE)
```

---

seqmpos   *Number of matching positions between two sequences.*

Description

Returns the number of common elements, i.e., same states appearing at the same position in the two sequences.

Usage

```
seqmpos(seq1, seq2, with.missing=FALSE)
```

Arguments

seq1   a sequence from a sequence object.

seq2   a sequence from a sequence object.

with.missing if TRUE, gaps appearing at the same position in both sequences are also considered as common elements.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for help page)

See Also

[seqLLCP](#), [seqLLCS](#) .

Examples

```
data(famform)
famform.seq <- seqdef(famform)

seqmpos(famform.seq[1,],famform.seq[2,])
seqmpos(famform.seq[2,],famform.seq[4,])

## Example with gaps in sequences
a <- c(NA,"A",NA,"B","C")
b <- c(NA,"C",NA,"B","C")

ex1.seq <- seqdef(rbind(a,b))

seqmpos(ex1.seq[1,], ex1.seq[2,])
seqmpos(ex1.seq[1,], ex1.seq[2,], with.missing=TRUE)
```

---

seqnum   *Transform into a sequence object with numerical alphabet.*

Description

The function seqnum transforms the provided state sequence object into an equivalent sequence object in which the original alphabet is replaced with an alphabet of numbers ranging from 0 to (nbstates-1).

Usage

```
 seqnum(seqdata, with.missing=FALSE)
```

Arguments

seqdata   a state sequence object as defined by the [seqdef](#) function.

with.missing   logical: Should missing elements in the sequences be turned into numerical values as well? The code for missing values in the sequences is retrieved from the 'nr' attribute of seqdata.

Details

The first state (for example 'A') is coded with the value 0, the second state (for example 'B') is coded with the value 1, etc... The function returns a sequence object containing the original sequences coded with the new numerical alphabet ranging from 0 to (nbstates-1)

Author(s)

Alexis Gabadinho

See Also

seqdef, alphabet

Examples

```
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## The first 10 sequences in the actcal.seq
## sequence object
actcal.seq[1:10,]
alphabet(actcal.seq)

## The first 10 sequences in the actcal.seq
## sequence object with numerical alphabet
seqnum(actcal.seq[1:10,])

## states A,B,C,D are now coded 0,1,2,3
alphabet(seqnum(actcal.seq))
```

---

seqpcplot   *Parallel coordinate plot for sequence data*

Description

A decorated parallel coordinate plot to render the order of the successive elements in sequences. The sequences are displayed as jittered frequency-weighted parallel lines. The plot is also embedded as the type="pc" option of the seqplot function and serves as plot method for seqe and seqelist objects.

Usage

```
seqpcplot(seqdata, group = NULL, weights = NULL,
          cex = 1, lwd = 1/4, cpal = NULL, grid.scale = 1/5,
          ltype = "unique", embedding = "most-frequent",
          lorder = NULL , lcourse = "upwards",
          filter = NULL, hide.col = "grey80",
          alphabet = NULL, missing = "auto", order.align = "first",
          title = NULL, xlab = NULL, ylab = NULL,
          xaxis = TRUE, yaxis = TRUE, axes = "all",
          xtlab = NULL, cex.plot = 1,
          rows = NA, cols = NA, plot = TRUE,
          seed = NULL, ...)

seqpcfilter(method = c("minfreq", "cumfreq", "linear"), level = 0.05)
```

Arguments

seqdata	The sequence data. Either an event sequence object of class seqelist (see [seqecreate](#)) or a state sequence object of class stslist (see [seqdef](#)).
group	a vector (numeric or factor) of group memberships of length equal the number of sequences. When specified, one plot is generated for each different membership value.
weights	a numeric vector of weights of length equal the number of sequences. Overrides weights in the seqdata object.
cex	expansion factor for the squared symbols.
lwd	expansion factor for line widths. The expansion is relative to the size of the squared symbols.
cpal	color palette vector for line coloring.
grid.scale	Expansion factor for the translation zones.
ltype	the type of sequence that is drawn. Either "unique" to render unique patterns or "non-embeddable" to render non-embeddable sequences.
embedding	The method for embedding sequences embeddable in multiple non-embeddable sequences. Either "most-frequent" (default) or "uniformly". Relevant only with ltype = "non-embeddable".
lorder	line ordering. Either "background" or "foreground".
lcourse	Method to connect simultaneous elements with the preceding and following ones. Either "upwards" (default) or "downwards".
filter	list of line coloring options. See details.
hide.col	Color for sequences filtered-out by the filter specification.
alphabet	a vector of response levels in the order they should appear on the y-axis. This argument is solely relevant for seqelist objects.
missing	character. Whether and how missing values should be displayed. Available are "auto", "show" and "hide". If "auto", the plot will show missings only if present. "hide" will fade out missings and "show" will always show missings.
order.align	Aligning method. For aligning on order positions use either "first" (default) or "last". Option "first" numbers the positions from the beginning while "last" numbers them from the end. With order.align = "time", the elements in the sequences are aligned on their rounded timestamps.
title	title for the graphic.
xlab	label for the x axis
ylab	label for the y axis
xaxis	logical: Should x-axis be plotted?
yaxis	logical: Should y-axis be plotted?
axes	if set as "all" (default value) x-axes are drawn for each plot in the graphic. If set as "bottom" and group is used, axes are drawn only under the plots at the bottom of the graphic area. If FALSE, no x-axis is drawn.
xtlab	labels for the x-axis ticks.

cex.plot	expansion factor for the size of the font for the axis labels and names. The default value is 1. Values lesser than 1 will reduce the size of the font, values greater than 1 will increase the size.
rows,cols	integers to arrange the plot panel design.
plot	logical. If `FALSE` nothing is plotted and an object of class seqpcplot is returned by default.
seed	integer. Start seed value.
method	character string. Defines the filtering function. Available are `"minfreq"`, `"cumfreq"` and `"linear"`.
level	numeric scalar between 0 and 1. The frequency threshold for the filtering methods `"minfreq"` and `"cumfreq"`.
...	arguments to be passed to other methods, such as graphical parameters (see [par](par)).

Details

For plots by groups specified with the `group` argument, plotted line widths and point sizes reflect relative frequencies within group.

The `filter` argument serves to specify filters to gray less interesting patterns. The filtered-out patterns are displayed in the `hide.col` color. The `filter` argument expects a list with at least elements `type` and `value`. The following types are implemented:

Type `"sequence"`: colors a specific pattern, for example assign
`filter = list(type = "sequence", value = "(Leaving   Home,Union)-(Child)")`.

Type `"subsequence"`: colors patterns which include a specific subsequence, for example
`filter = list(type =   "subsequence", value = "(Child)-(Marriage)")` .

Type `"value"`: gradually colors the patterns according to the numeric vector (of length equal to the number of sequences) provided as `"value"` element in the list. You can give something like `filter = list(type = "value", value = c(0.2, 1, ...))` or provide the distances to the medoid as `value` vector for example.

Type `"function"`: colors the patterns depending on the values returned by a [0,1] valued function of the frequency x of the pattern. Three native functions can be used: `"minfreq"`, `"cumfreq"` and `"linear"`. Use `filter = list(type =   "function", value = "minfreq", level = 0.05)` to color patterns with a support of at least 5% (within group). Use
`filter  =  list(type        =  "function",  value  =  "cumfreq",  level  =  0.5)` to highlight the 50% most frequent patterns (within group). Or, use `filter  =  list(type  = "function",  value  =  "linear")` to use a linear gradient for the color intensity (the most most frequent trajectory obtains 100% intensity). Other user-specified functions can be provided by giving something like
`filter = list(type = "function", value =   function(x, arg1, arg2) {return(x / max(x) * arg1 / arg2)},` This latter function adjusts gradually the color intensity of patterns according to the frequency of the pattern.

The function `seqpcfilter` is a convenience function for type `"function"`. The three examples above can be imitated by `seqpcfilter("minfreq", 0.05)`, `seqpcfilter("cumfreq",   0.5)` and `seqpcfilter("linear")`.

If a numeric scalar is assigned to `filter`, the `"minfreq"` filter is used.

Value

seqpcplot returns an object of class "seqpcplot" with various information for constructing the plot, e.g. coordinates. There is also a summary method for such objects.

Author(s)

Reto Bürgin (with Gilbert Ritschard for the help page)

References

Bürgin, R. and G. Ritschard (2014), A decorated parallel coordinate plot for categorical longitudinal data, *The American Statistician* 68(2), 98-103.

See Also

[seqplot](), [seqdef](), [seqecreate]()

Examples

```
## ================
## plot biofam data
## ================

data(biofam)
lab <- c("Parent","Left","Married","Left+Marr","Child","Left+Child",
         "Left+Marr+Child","Divorced")

## plot state sequences in STS representation
## ==========================================

## creating the weighted state sequence object.
biofam.seq <- seqdef(data = biofam[,10:25], labels = lab,
                     weights = biofam$wp00tbgs)

## select the first 20 weighted sequences (sum of weights = 18)
biofam.seq <- biofam.seq[1:20, ]

par(mar=c(4,8,2,2))
seqpcplot(seqdata = biofam.seq, order.align = "time")

## .. or
seqplot(seqdata = biofam.seq, type = "pc", order.align = "time")

## Distinct successive states (DSS)
## ==========================================

seqplot(seqdata = biofam.seq, type = "pc", order.align = "first")

## .. or (equivalently)

biofam.DSS <- seqdss(seqdata = biofam.seq) # prepare format
```

```
seqpcplot(seqdata = biofam.DSS)

## plot TSE data converted from state sequences
## =============================================

## conversion
biofam.TSE <- seqformat(data = biofam.seq, from = "STS", to = "TSE",
                        tevent = seqetm(biofam.seq, method = "state"))
biofam.TSE$event <- factor(biofam.TSE$event, levels = lab) # define alphabet
biofam.TSE$time <- biofam.TSE$time + 15 # correct age

seqpcplot(seqdata = biofam.TSE, order.align = "time")


## plot event sequences
## ====================

biofam.seqe <- seqecreate(biofam.seq, tevent = "state") # prepare data

## plot the time in the x-axis
seqpcplot(seqdata = biofam.seqe, order.align = "time", alphabet = lab)

## ordering of events
seqpcplot(seqdata = biofam.seqe, order.align = "first", alphabet = lab)

## ... or
plot(biofam.seqe, order.align = "first", alphabet = lab)

## additional arguments
## ====================

## non-embeddable sequences
seqpcplot(seqdata = biofam.seqe, ltype = "non-embeddable",
          order.align = "first", alphabet = lab)

## align on last event
par(mar=c(4,8,2,2))
seqpcplot(seqdata = biofam.seqe, order.align = "last", alphabet = lab)

## use group variables
seqpcplot(seqdata = biofam.seqe, group = biofam$sex[1:20],
          order.align = "first", alphabet = lab)

## color patterns (Parent)-(Married) and (Parent)-(Left+Marr+Child)
par(mfrow = c(1, 1))
seqpcplot(seqdata = biofam.seqe,
          filter = list(type = "sequence",
                        value=c("(Parent)-(Married)",
                                "(Parent)-(Left+Marr+Child)")),
          alphabet = lab, order.align = "first")

## color subsequence pattern (Parent)-(Left)
seqpcplot(seqdata = biofam.seqe,
```

```
              filter = list(type = "subsequence",
                             value = "(Parent)-(Left)"),
              alphabet = lab, order.align = "first")

## color sequences over 10% (within group) (function method)
seqpcplot(seqdata = biofam.seqe,
          filter = list(type = "function",
                         value = "minfreq",
                         level = 0.1),
          alphabet = lab, order.align = "first", seed = 1)

## .. same result using the convenience functions
seqpcplot(seqdata = biofam.seqe,
          filter = 0.1,
          alphabet = lab, order.align = "first", seed = 1)

seqpcplot(seqdata = biofam.seqe,
          filter = seqpcfilter("minfreq", 0.1),
          alphabet = lab, order.align = "first", seed = 1)

## highlight the 50% most frequent sequences
seqpcplot(seqdata = biofam.seqe,
          filter = list(type = "function",
                         value = "cumfreq",
                         level = 0.5),
          alphabet = lab, order.align = "first", seed = 2)

## .. same result using the convenience functions
seqpcplot(seqdata = biofam.seqe,
          filter = seqpcfilter("cumfreq", 0.5),
          alphabet = lab, order.align = "first", seed = 2)

## linear gradient
seqpcplot(seqdata = biofam.seqe,
          filter = list(type = "function",
                         value = "linear"),
          alphabet = lab, order.align = "first", seed = 2)

seqpcplot(seqdata = biofam.seqe,
          filter = seqpcfilter("linear"),
          alphabet = lab, order.align = "first", seed = 1)
```

seqplot   *Plot state sequence objects*

Description

High level plot functions for state sequence objects that can produce state distribution (chronograms), frequency, index, transversal entropy, sequence of modes, meant time, and representative plots.

Usage

```
seqplot(seqdata, group=NULL, type="i", title=NULL,
        cpal=NULL, missing.color=NULL,
        ylab=NULL, yaxis=TRUE, axes="all", xtlab=NULL, cex.plot=1,
        withlegend="auto", ltext=NULL, cex.legend=1,
        use.layout=(!is.null(group) | withlegend!=FALSE),
        legend.prop=NA, rows=NA, cols=NA, ...)

seqdplot(seqdata, group=NULL, title=NULL, ...)
seqfplot(seqdata, group=NULL, title=NULL, ...)
seqiplot(seqdata, group=NULL, title=NULL, ...)
seqIplot(seqdata, group=NULL, title=NULL, ...)
seqHtplot(seqdata, group=NULL, title=NULL, ...)
seqmsplot(seqdata, group=NULL, title=NULL, ...)
seqmtplot(seqdata, group=NULL, title=NULL, ...)
```

Arguments

seqdata	a state sequence object created with the [seqdef](#) function.
group	Plots one plot for each level of the factor given as argument.
type	the type of the plot. Available types are "d" for state distribution plots (chronograms), "f" for sequence frequency plots, "Ht" for transversal entropy plots, "i" for selected sequence index plots, "I" for whole set index plots, "ms" for plotting the sequence of modal states, "mt" for mean times plots, "pc" for parallel coordinate plots and "r" for representative sequence plots.
title	title for the graphic. Default is NULL.
cpal	Color palette used for the states. By default, the cpal attribute of the seqdata sequence object is used (see [seqdef](#)). If user specified, a vector of colors with number of elements equal to the number of distinct states.
missing.color	alternative color for representing missing values inside the sequences. By default, this color is taken from the missing.color attribute of the plotted sequence object.
ylab	an optional label for the y-axis. If set to NA, no label is drawn.
yaxis	controls whether a y-axis is plotted. When set to TRUE (default value), sequence indexes are displayed for "i" and "I", mean time values for "mt" and percentages for "d" and "f".
axes	if set to "all" (default value) x axes are drawn for each plot in the graphic. If set to "bottom" and group is used, axes are drawn only under the plots located at the bottom of the graphic area. If FALSE, no x-axis is drawn.

xtlab	optional labels for the x-axis tick labels. If unspecified, the column names of the seqdata sequence object are used (see [seqdef](#)).
cex.plot	expansion factor for setting the size of the font for the axis labels and names. The default value is 1. Values lesser than 1 will reduce the size of the font, values greater than 1 will increase the size.
withlegend	defines if and where the legend of the state colors is plotted. The default value "auto" sets the position of the legend automatically. Other possible value is "right". Obsolete value TRUE is equivalent to "auto".
ltext	optional description of the states to appear in the legend. Must be a vector of character strings with number of elements equal to the size of the alphabet. If unspecified, the label attribute of the seqdata sequence object is used (see [seqdef](#)).
cex.legend	expansion factor for setting the size of the font for the labels in the legend. The default value is 1. Values lesser than 1 will reduce the size of the font, values greater than 1 will increase the size.
use.layout	if TRUE, [layout](#) is used to arrange plots when using the group option or plotting a legend. When layout is activated, the standard ['par](#)(mfrow=....)' for arranging plots does not work. With withlegend=FALSE and group=NULL, layout is automatically deactivated and 'par(mfrow=....)' can be used.
legend.prop	sets the proportion of the graphic area used for plotting the legend when use.layout=TRUE and withlegend=TRUE. Default value is set according to the place (bottom or right of the graphic area) where the legend is plotted. Values from 0 to 1.
rows,cols	optional arguments to arrange plots when use.layout=TRUE.
...	arguments to be passed to the function called to produce the appropriate statistics and the associated plot method (see details), or other graphical parameters. For example the weighted argument can be passed to control whether (un)weighted statistics are produced or with.missing argument to take missing values into account when computing transversal or longitudinal state distributions.

Details

seqplot is the generic function for high level plots of state sequence objects with group splits and automatic display of the color legend. Many different types of plots can be produced by means of the type argument. Except for sequence index plots, seqplot first calls the specific function producing the required statistics and then the plot method for objects produced by this function (see below). For sequence index plots, the state sequence object itself is plotted by calling the [plot.stslist](#) method. When splitting by groups and/or displaying the color legend, the [layout](#) function is used for arranging the plots.

The seqdplot, seqfplot, seqiplot, seqIplot, seqHtplot, seqmsplot, seqmtplot, seqpcplot and seqrplot functions are aliases for calling seqplot with type argument set respectively to "d", "f", "i", "I", "Ht", "ms", "mt", "pc" or "r".

State distribution plot (type="d") represent the sequence of the cross-sectional state frequencies by position (time point) computed by the [seqstatd](#) function. Such plots are also known as chronograms.

Sequence frequency plots (type="f") display the most frequent sequences, each one with an horizontal stack bar of its successive states. Sequences are displayed bottom-up in decreasing order

of their frequencies (computed by the `seqtab` function). The `plot.stslist.freq` plot method is called for producing the plot.

The `tlim` optional argument may be specified for selecting the sequences to be plotted (default is 1:10, i.e. the 10 most frequent sequences). The width of the bars representing the sequences is by default proportional to their frequencies, but this can be disabled with the `pbarw=FALSE` optional argument. If weights have been specified when creating seqdata, weighted frequencies will be returned by seqtab since the default option is `weighted=TRUE`. See examples below, the `seqtab` and `plot.stslist.freq` manual pages for a complete list of optional arguments and *Müller et al., (2008)* for a description of sequence frequency plots.

In *sequence index plots* (`type="i"` or `type="I"`), the requested individual sequences are rendered with horizontal stacked bars depicting the states over successive positions (time). Optional arguments are `tlim` for specifying the indexes of the sequences to be plotted (when `type="i"` defaults to the first ten sequences, i.e `tlim=1:10`). For plotting nicely a (big) whole set one can use `type="I"` which is the same as using `tlim=0` together with the additional graphical parameters `border=NA` and `space=0` to suppress bar borders and space between bars. The `sortv` argument can be used to pass a vector of numerical values for sorting the sequences or to specify a sorting method. See `plot.stslist` for a complete list of optional arguments and their description.

The interest of sequence index plots has, for instance, been stressed by *Scherer (2001)* and *Brzinsky-Fay et al. (2006)*. Notice that index plots for thousands of sequences result in very heavy PDF or POSTSCRIPT graphic files. Dramatic file size reduction may be achieved by saving the figures in bitmap format with using for instance the `png` graphic device instead of `postscript` or `pdf`.

The *transversal entropy plot* (`type="Ht"`) displays the evolution over positions of the transversal entropies (*Billari, 2001*). Transversal entropies are computed by calling `seqstatd` function and then plotted by calling the `plot.stslist.statd` plot method.

The *modal state sequence plot* (`type="ms"`) displays the sequence of the modal states with each mode proportional to its frequency at the given position. The `seqmodst` function is called which returns the sequence and the result is plotted by calling the `plot.stslist.modst` plot method.

The *mean time plot* (`type="mt"`) displays the mean time spent in each state of the alphabet as computed by the `seqmeant` function. The `plot.stslist.meant` plot method is used to plot the resulting statistics. Set `serr=TRUE` to display error bars on the mean time plot.

The *representative sequence plot* (`type="r"`) displays a reduced, non redundant set of representative sequences extracted from the provided state sequence object and sorted according to a representativeness criterion. The `seqrep` function is called to extract the representative set which is then plotted by calling the `plot.stslist.rep` method. A distance matrix is required that is passed with the `dist.matrix` argument or by calling the seqdist function if `dist.matrix=NULL`. The `criterion` argument sets the representativeness criterion used to sort the sequences. See examples below, the `seqrep` and `plot.stslist.rep` manual pages for a complete list of optional arguments and *Gabadinho et al. (2009)* for more details on the extraction of representative sets.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

References

Billari, F. C. (2001). The analysis of early life courses: Complex description of the transition to adulthood. *Journal of Population Research* **18**(2), 119-142.

Brzinsky-Fay C., U. Kohler, M. Luniak (2006). Sequence Analysis with Stata. *The Stata Journal*, **6**(4), 435-460.

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

Gabadinho A, Ritschard G, Studer M, Müller NS (2011). "Extracting and Rendering Representative Sequences", In A Fred, JLG Dietz, K Liu, J Filipe (eds.), *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 128 of *Communications in Computer and Information Science (CCIS)*, pp. 94-106. Springer-Verlag.

Müller, N. S., A. Gabadinho, G. Ritschard and M. Studer (2008). Extracting knowledge from life courses: Clustering and visualization. In *Data Warehousing and Knowledge Discovery, 10th International Conference DaWaK 2008, Turin, Italy, September 2-5*, LNCS 5182, Berlin: Springer, 176-185.

Scherer S (2001). Early Career Patterns: A Comparison of Great Britain and West Germany. *European Sociological Review*, **17**(2), 119-144.

See Also

plot.stslist.statd, plot.stslist.freq, plot.stslist, plot.stslist.modst, plot.stslist.meant, plot.stslist.rep seqpcplot, seqrplot .

Examples

```
## =======================================================
## Creating state sequence objects from example data sets
## =======================================================

## biofam data set
data(biofam)
## We use only a sample of 300 cases
set.seed(10)
biofam <- biofam[sample(nrow(biofam),300),]
biofam.lab <- c("Parent", "Left", "Married", "Left+Marr",
                "Child", "Left+Child", "Left+Marr+Child", "Divorced")
biofam.seq <- seqdef(biofam, 10:25, labels=biofam.lab)

## actcal data set
data(actcal)
## We use only a sample of 300 cases
set.seed(1)
actcal <- actcal[sample(nrow(actcal),300),]
actcal.lab <- c("> 37 hours", "19-36 hours", "1-18 hours", "no work")
actcal.seq <- seqdef(actcal,13:24,labels=actcal.lab)

## ex1 using weights
data(ex1)
ex1.seq <- seqdef(ex1, 1:13, weights=ex1$weights)

## ========================
## Sequence frequency plots
## ========================
```

```
## Plot of the 10 most frequent sequences
seqplot(biofam.seq, type="f")

## Grouped by sex
seqfplot(actcal.seq, group=actcal$sex)

## Unweighted vs weighted frequencies
seqfplot(ex1.seq, weighted=FALSE)
seqfplot(ex1.seq, weighted=TRUE)


## =====================
## Modal states sequence
## =====================
seqplot(biofam.seq, type="ms")
## same as
seqmsplot(biofam.seq)


## ====================
## Representative plots
## ====================


## Computing a distance matrix
## with OM metric
costs <- seqsubm(biofam.seq, method="TRATE")
biofam.om <- seqdist(biofam.seq, method="OM", sm=costs)

## Plot of the representative sets grouped by sex
## using the default density criterion
seqrplot(biofam.seq, group=biofam$sex, dist.matrix=biofam.om)

## Plot of the representative sets grouped by sex
## using the "dist" (centrality) criterion
seqrplot(biofam.seq, group=biofam$sex, criterion="dist", dist.matrix=biofam.om)


## ====================
## Sequence index plots
## ====================


## First ten sequences
seqiplot(biofam.seq)

## All sequences sorted by age in 2000
## grouped by sex
## using 'border=NA' and 'space=0' options to have a nicer plot
seqiplot(actcal.seq, group=actcal$sex, tlim=0, border=NA, space=0,
        sortv=actcal$age00)


## =======================
## State distribution plot
## =======================
```

```
## biofam grouped by sex
seqplot(biofam.seq, type="d", group=actcal$sex)

## actcal grouped by sex
seqplot(actcal.seq, type="d", group=actcal$sex)

## ===================
## Cross-sectional entropy plot
## ===================
seqplot(biofam.seq, type="Ht", group=biofam$sex)

## ===============
## Meant time plot
## ===============

## actcal data set, grouped by sex
seqplot(actcal.seq, type="mt", group=actcal$sex)

## biofam data set, grouped by sex
seqmtplot(biofam.seq, group=biofam$sex)
```

---

seqpm   *Find substring patterns in sequences*

Description

Search for a pattern (substring) into sequences.

Usage

```
seqpm(seqdata, pattern, sep="")
```

Arguments

seqdata	a sequence object as defined by the [seqdef](#) function.
pattern	a character string representing the pattern (substring) to search for.
sep	state separator used in the pattern definition.

Details

This function searches a pattern (a character string) into a set of sequences and returns the results as a list with two elements: 'Nbmatch' the number of occurrences of the pattern and 'MatchesIndex' the vector of indexes (row numbers) of the sequences that match the pattern (see examples below).

Value

a list with two elements (see details).

Author(s)

Alexis Gabadinho

Examples

```
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## search for pattern "DAAD"
## (no work-full time work-full time work-no work)
## results are stored in the 'daad' object
daad <- seqpm(actcal.seq,"DAAD")

## Looking at the sequences
## containing the pattern
actcal.seq[daad$MIndex,]

## search for pattern "AD"
## (full time work-no work)
seqpm(actcal.seq,"AD")
```

---

seqrecode   *Recoding state sequence objects and factors*

Description

Utilities for recoding factors or state sequence objects created with seqdef.

Usage

```
seqrecode(seqdata, recodes, otherwise = NULL,
    labels = NULL, cpal = NULL)
recodef(x, recodes, otherwise=NULL, na=NULL)
```

Arguments

seqdata   The state sequence object to be recoded (created with seqdef).

recodes   A list specifying the recoding operations where each element is in the form
             newcode=oldcode or newcode=c(oldcode1, oldcode2,...). The rules are
             treated in the same order as they appear, hence subsequent rules may modify the
             first ones.

otherwise   NULL or Character. Level given to cases uncovered by the recodes list. If NULL,
             old states remain unchanged.

labels   optional state labels used for the color legend of TraMineR's graphics. If NULL
             (default), the state names in the alphabet are also used as state labels (see seqdef).

cpal	an optional color palette for representing the newly defined alphabet in graphics. If NULL (default), a color palette is created from the colors in seqdata by assigning to newcode the color of the first old state listed as oldcode and by leaving the colors of the other states unchanged.
x	A factor to be recoded.
na	Character vector. If not NULL, the list of states that should be recoded as NA (missing values).

Value

The recoded factor or state sequence object.

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

[seqdef](#) to create a state sequence object.

Examples

```
## Recoding a state sequence object with seqrecode
data(actcal)
## Creating a state sequence object
actcal.seq <- seqdef(actcal,13:24, labels=c("> 37 hours", "19-36 hours",
   "1-18 hours", "no work"))
## Regrouping states B and C and setting the whole alphabet to A BC D
actcal.new <-seqrecode(actcal.seq,
    recodes = list("A"="A", "BC"=c("B", "C"), "D"="D"))
## Crosstabulate the first column of the recoded and
## original state sequence objects
table(actcal.new[,1], actcal.seq[,1])

## Same as before but using automatically original
## codes for unspecified states.
actcal.new2 <-seqrecode(actcal.seq,
recodes = list("BC"=c("B", "C")))
table(actcal.new2[,1], actcal.seq[,1])

## Same as before but using otherwise
actcal.new3 <-seqrecode(actcal.seq, recodes = list("A"="A", "D"="D"),
   otherwise="BC")
table(actcal.new3[,1], actcal.seq[,1])

## Recoding factors
## Recoding the marital status to oppose married to all other case
maritalstatus <- recodef(actcal$civsta00,
recodes=list("Married"="married"), otherwise="Single")
summary(maritalstatus)
table(maritalstatus, actcal$civsta00)
```

```
## Recoding the number of kids in the household
## -2 is a missing value
nbkids <- recodef(actcal$nbkid00,
    recodes=list("None"=0, "One"=1, "Two or more"=2:10), na=-2)
table(nbkids, actcal$nbkid00, useNA="always")
```

---

seqrep   *Extracting sets of representative sequences*

Description

Returns either an as small as possible set of non redundant representatives covering (having in their neighborhood) a desired percentage of all sequences, or a given number of patterns with highest coverage. Special cases are single representatives such as the medoid or the sequence pattern with densest neighborhood. See [plot.stslist.rep](#) for the plot method and [seqplot](#) for other plot options.

Usage

```
  seqrep(seqdata, criterion="density", score=NULL,
    decreasing=TRUE, trep=0.25, nrep=NULL,
    tsim=0.1, dmax=NULL, dist.matrix=NULL, weighted=TRUE, ...)
```

Arguments

seqdata	a state sequence object as defined by the [seqdef](#) function.
criterion	the representativeness criterion for sorting the candidate list. One of "freq" (sequence frequency), "density" (neighborhood density), "mscore" (mean state frequency), "dist" (centrality) and "prob" (sequence likelihood). See details.
score	an optional vector of representativeness scores for sorting the sequences in the candidate list. The length of the vector must be equal to the number of sequences in the sequence object.
decreasing	if a score vector is provided, indicates whether the objects in the candidate list must be sorted in ascending or descending order of this score. Default is TRUE, i.e. descending. The first object in the candidate list is then supposed to be the most representative.
trep	coverage threshold, i.e., minimum proportion of sequences that should have a representative in their neighborhood (neighborhood radius is defined by tsim).
nrep	number of representative sequences. If NULL (default), the size of the representative set is controlled by trep.
tsim	neighborhood radius as a percentage of the maximum (theoretical) distance dmax. Defaults to 0.1 (10%). Sequence y is redundant to sequence x when it is in the neighborhood of x, i.e., within a distance tsim*dmax from x.

dmax   maximum theoretical distance. Used to derive the neighborhood radius as `tsim*dmax`. If `NULL`, the value of dmax is derived from the dissimilarity matrix.

dist.matrix   matrix of pairwise dissimilarities between sequences in seqdata. If `NULL`, the matrix is computed by calling the [seqdist](#) function. In that case, optional arguments to be passed to the seqdist function (see `...` hereafter) should also be provided.

weighted   logical: Should weights assigned to the state sequence object be accounted for? (See [seqdef](#).) Set as `FALSE` to ignore the weights.

...   optional arguments to be passed to the seqdist function, mainly `dist.method` specifying the metric for computing the distance matrix, `norm` for normalizing the distances, `indel` and `sm` for indel and substitution costs when Optimal Matching metric is chosen. See [seqdist](#) manual page for details.

Details

The representative set is obtained by an heuristic. Representatives are selected by successively extracting from the sequences sorted by their representativeness score those which are not redundant with already retained representatives. The selection stops when either the desired coverage or the wanted number of representatives is reached. Sequences are sorted either by the values provided as score argument or by specifying one of the following as criterion argument: `"freq"` (*sequence frequency*), `"density"` (*neighborhood density*), `"mscore"` (*mean state frequency*), `"dist"` (*centrality*) and `"dist"` (*sequence likelihood*).

With the *sequence frequency* criterion, the more frequent a sequence the more representative it is supposed to be. Therefore, sequences are sorted in decreasing frequency order.

The *neighborhood density* is the number—density—of sequences in the neighborhood of the sequence. This requires to set the neighborhood radius tsim. Sequences are sorted in decreasing density order.

The *mean state frequency* criterion is the mean value of the transversal frequencies of the successive states. Let $s = s_1 s_2 \cdots s_\ell$ be a sequence of length ℓ and $(f_{s_1}, f_{s_2}, \ldots, f_{s_\ell})$ the frequencies of the states at (time-)position $(t_1, t_2, \ldots t_\ell)$. The mean state frequency is the sum of the state frequencies divided by the sequence length

$$MSF(s) = \frac{1}{\ell} \sum_{i=1}^{\ell} f_{s_i}$$

The lower and upper boundaries of MSF are 0 and 1. MSF is equal to 1 when all the sequences in the set are identical, i.e. when there is a single sequence pattern. The most representative sequence is the one with the highest score.

The *centrality* criterion is the sum of distances to all other sequences. The smallest the sum, the most representative the sequence.

The *sequence likelihood* $P(s)$ is defined as the product of the probability with which each of its observed successive state is supposed to occur at its position. Let $s = s_1 s_2 \cdots s_\ell$ be a sequence of length ℓ. Then

$$P(s) = P(s_1, 1) \cdot P(s_2, 2) \cdots P(s_\ell, \ell)$$

with $P(s_t, t)$ the probability to observe state s_t at position t.
The question is how to determinate the state probabilities $P(s_t, t)$. One commonly used method

for computing them is to postulate a Markov Chain model, which can be of various order. The implemented criterion considers the probabilities derived from the first order Markov model, that is each $P(s_t, t)$, $t > 1$ is set to the transition rate $p(s_t|s_{t-1})$ estimated across sequences from the observations at positions t and $t - 1$. For $t = 1$, we set $P(s_1, 1)$ to the observed frequency of the state s_1 at position 1.

The likelihood $P(s)$ being generally very small, we use $-\log P(s)$ as sorting criterion. The latter quantity reaches its minimum for $P(s)$ equal to 1, which leads to sort the sequences in ascending order of their score.

Use `criterion="dist"` and `nrep=1` to get the medoid and `criterion="density"` and `nrep=1` to get the densest sequence pattern.

For more details, see *Gabadinho & Ritschard, 2013*.

Value

An object of class `stslist.rep`. This is actually a state sequence object (containing a list of state sequences) with the following additional attributes:

Scores	a vector with the representative score of each sequence in the original set given the chosen criterion.
Distances	a matrix with the distance of each sequence to its nearest representative.
Statistics	a data frame with quality measures for each representative sequence: number of sequences attributed to the representative, number of sequence in the representative's neighborhood, mean distance to the representative.
Quality	overall quality measure.

Print, plot and summary methods are available. More elaborated plots are produced by the seqplot function using the `type="r"` argument, or the seqrplot alias.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

References

Gabadinho A, Ritschard G (2013). "Searching for typical life trajectories applied to child birth histories", In R Lévy, E. Widmer (eds.), *Gendered Life Courses*, pp. 287-312. Vienna: LIT.

Gabadinho A, Ritschard G, Studer M, Müller NS (2011). "Extracting and Rendering Representative Sequences", In A Fred, JLG Dietz, K Liu, J Filipe (eds.), *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 128 of *Communications in Computer and Information Science (CCIS)*, pp. 94-106. Springer-Verlag.

See Also

seqplot, plot.stslist.rep, dissrep, disscenter

Examples

```
## Defining a sequence object with the data in columns 10 to 25
## (family status from age 15 to 30) in the biofam data set
data(biofam)
biofam.lab <- c("Parent", "Left", "Married", "Left+Marr",
"Child", "Left+Child", "Left+Marr+Child", "Divorced")
biofam.seq <- seqdef(biofam, 10:25, labels=biofam.lab)

## Computing the distance matrix
costs <- seqsubm(biofam.seq, method="TRATE")
biofam.om <- seqdist(biofam.seq, method="OM", sm=costs)

## Representative set using the neighborhood density criterion
biofam.rep <- seqrep(biofam.seq, dist.matrix=biofam.om, criterion="density")
biofam.rep
summary(biofam.rep)
plot(biofam.rep)
```

---

seqsep   *Adds separators to sequences stored as character string*

Description

Adds separators to sequences stored as character string.

Usage

```
seqsep(seqdata, sl=1, sep="-")
```

Arguments

seqdata	a dataframe or matrix containing sequence data, as vectors of states or events.
sl	the length of the states (the number of characters used to represent them). Default is 1.
sep	the character used as separator. Set by default as "-".

See Also

[seqdecomp](#).

Examples

```
seqsep("ABAAAAAAD")
```

---

seqST	*Sequences turbulence*

Description

Computes Elzinga's turbulence for each sequence in a sequence data set.

Usage

```
seqST(seqdata)
```

Arguments

seqdata   a state sequence object as returned by the the [seqdef](#) function.

Details

Sequence turbulence is a measure proposed by *Elzinga & Liefbroer (2007)*. It is based on the number $\phi(x)$ of distinct subsequences that can be extracted from the distinct successive state sequence and the variance of the consecutive times t_i spent in the distinct states. For a sequence x, the formula is

$$T(x) = \log_2(\phi(x) \frac{s_{t,max}^2(x) + 1}{s_t^2(x) + 1})$$

where $s_t^2(x)$ is the variance of the successive state durations in sequence x and $s_{t,max}^2(x)$ is the maximum value that this variance can take given the total duration of the sequence. This maximum is computed as

$$s_{t,max}^2 = (d - 1)(1 - \bar{t})^2$$

where $\bar{t}$ is the mean consecutive time spent in the distinct states, i.e. the sequence duration divided by the number d of distinct states in the sequence.

The function searches for missing states in the sequences and if found, adds the missing state to the alphabet for the computation of the turbulence. In this case the [seqdss](#) and [seqdur](#) functions for extracting the distinct successive state sequences and the associated durations are called with the {with.missing=TRUE} argument. A missing state in a sequence is considered as the occurence of an additional symbol of the alphabet, and two or more consecutive missing states are considered as two or more occurences of the same state. Hence the DSS of A-A-*-*-*-B-B-C-C-D is A-*-B-C-D and the associated durations are 2-3-2-2-1.

Value

a vector of length equal to the number of sequences in seqdata containing the turbulence value of each sequence.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

References

Elzinga, Cees H. and Liefbroer, Aart C. (2007). De-standardization of Family-Life Trajectories of Young Adults: A Cross-National Comparison Using Sequence Analysis. *European Journal of Population*, 23, 225-250.

See Also

seqdss, seqdur. For another composite measure of sequence complexity see and seqici.

Examples

```
## Loading the 'actcal' example data set
data(actcal)

## Defining a sequence object with data in columns 13 to 24
## (activity status from january to december 2000)
actcal.seq <- seqdef(actcal,13:24, informat='STS')

## Computing the sequences turbulence
turb <- seqST(actcal.seq)

## Histogram for the turbulence
hist(turb)
```

---

seqstatd   *Sequence of transversal state distributions and their entropies*

Description

Returns the state frequencies, the number of valid states and the entropy of the state distribution at each position in the sequence.

Usage

```
seqstatd(seqdata, weighted=TRUE, with.missing=FALSE, norm=TRUE)
```

Arguments

seqdata	a state sequence object as defined by the seqdef function.
weighted	if TRUE, distributions account for the weights assigned to the state sequence object (see seqdef). Set as FALSE if you want ignore the weights.
with.missing	If FALSE (default value), returned distributions ignore missing values.
norm	if TRUE (default value), entropy is normalized, ie divided by the entropy of the alphabet. Set as FALSE if you want the entropy without normalization.

Details

In addition to the state distribution at each position in the sequence, the seqstatd function provides also for each time point the number of valid states and the Shannon entropy of the observed state distribution. Letting p_i denote the proportion of cases in state i at the considered time point, the entropy is

$$h(p_1, \ldots, p_s) = -\sum_{i=1}^{s} p_i \log(p_i)$$

where s is the size of the alphabet. The log is here the natural (base e) logarithm. The entropy is 0 when all cases are in the same state and is maximal when the same proportion of cases are in each state. The entropy can be seen as a measure of the diversity of states observed at the considered time point. An application of such a measure (but with aggregated transversal data) can be seen in *Billari (2001)* and *Fussell (2005).*

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

References

Billari, F. C. (2001). The analysis of early life courses: complex descriptions of the transition to adulthood. *Journal of Population Research* 18 (2), 119-24.

Fussell, E. (2005). Measuring the early adult life course in Mexico: An application of the entropy index. In R. Macmillan (Ed.), *The Structure of the Life Course: Standardized? Individualized? Differentiated?*, Advances in Life Course Research, Vol. 9, pp. 91-122. Amsterdam: Elsevier.

See Also

plot.stslist.statd the plot method for objects of class stslist.statd,
seqdplot for higher level plot of transversal distributions and
seqHtplot for plotting the transversal entropy over sequence positions.

Examples

```
data(biofam)
biofam.seq <- seqdef(biofam,10:25)
sd <- seqstatd(biofam.seq)
## Plotting the state distribution
plot(sd, type="d")

## Plotting the entropy indexes
plot(sd, type="Ht")

## ===================
## example with weights
## ===================
data(ex1)
ex1.seq <- seqdef(ex1, 1:13, weights=ex1$weights)

## Unweighted
```

```
seqstatd(ex1.seq, weighted=FALSE)

seqstatd(ex1.seq, weighted=TRUE)
```

---

seqstatf *State frequencies in the whole sequence data set*

Description

Overall frequency of each state of the alphabet in the state sequence object.

Usage

```
seqstatf(seqdata, weighted = TRUE)
```

Arguments

seqdata   a sequence object as defined by the [seqdef](#) function.

weighted   Logical. Should frequencies account for weights when present in the state se-
               quence object (see [seqdef](#)). Default is TRUE. If no weights were assigned during
               the creation of the sequence object, weighted=TRUE will yield the same result
               as weighted=FALSE since each sequence is allowed a weight of 1.

Details

The seqstatf function computes the (weighted) count and frequency of each state of the alphabet
in seqdata, i.e., the (weighted) sum of the occurrences of a state in seqdata.

Value

A data frame with as many rows as states in the alphabet and two columns, one for the count (Freq)
and one for the percentage frequencies (Percent).

Author(s)

Alexis Gabadinho

See Also

[seqstatd](#) for the state distribution by time point (position), [seqistatd](#) for the state distribution
within each sequence.

Examples

```
## Creating a sequence object from the actcal data set
data(actcal)
actcal.lab <- c(">37 hours", "19-36 hours", "1-18 hours", "no work")
actcal.seq <- seqdef(actcal, 13:24, labels=actcal.lab)

## States frequencies
seqstatf(actcal.seq)

## Example with weights
data(ex1)
ex1.seq <- seqdef(ex1, 1:13, weights=ex1$weights)

## Unweighted
seqstatf(ex1.seq, weighted=FALSE)

## Weighted
seqstatf(ex1.seq, weighted=TRUE)
```

---

seqstatl *List of distinct states or events (alphabet) in a sequence data set.*

Description

Returns a list containing distinct states or events found in a data frame or matrix containing sequence data, the alphabet.

Usage

```
seqstatl(data, var=NULL, format='STS')
```

Arguments

data	a data frame or matrix containing sequence data.
var	the list of columns containing the sequences. Default NULL means all columns. Whether the sequences are in the compressed (character strings) or extended format is automatically detected from the number of columns..
format	the format of the sequence data set. One of "STS", "SPS", "DSS". Default is "STS". The seqstatl function uses the [seqformat](#) function to translate between formats when necessary.

Author(s)

Alexis Gabadinho

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2009). Mining Sequence Data in R with the TraMineR package: A user's guide. Department of Econometrics and Laboratory of Demography, University of Geneva.

See Also

seqformat

Examples

```
data(actcal)
seqstatl(actcal,13:24)
```

---

seqsubm   *Create a substitution-cost matrix*

Description

The substitution-cost matrix is used when computing distances between sequences by the method of optimal matching. The function creates the substitution matrix using either a constant or the transition rates computed from the sequence data or other methods to be implemented in the future.

Usage

```
seqsubm(seqdata, method, cval=NULL, with.missing=FALSE,
        miss.cost=NULL, time.varying=FALSE, weighted=TRUE,
transition="both", lag=1, missing.trate=FALSE)
```

Arguments

seqdata	a sequence object as returned by the seqdef function.
method	method to compute transition rates. At this time, the methods available are constant value (method=″CONSTANT″) or substitution costs using transition rates (method=″TRATE″)
cval	the constant substitution cost if method ″CONSTANT″ is chosen. For method ″TRATE″, the base value from which transition probabilities are subtracted. If NULL, cval=2, unless transition is set to ″both″ and time.varying is TRUE in which case cval=4.
with.missing	if TRUE, an additional entry is added in the matrix for the missing states. Hence, a new "missing" state is added to the list of "valid" states. Use this if you want to compute distances with missing values inside the sequences. See *Gabadinho et al. (2010)* for more details on the options for handling missing values when computing distances between sequences.

miss.cost	the substitution cost for the missing state. The default set it to cval
time.varying	Logical. If TRUE return an [array](#) containing a distinct matrix for each time unit. The time is the third dimension (subscript).
weighted	Logical. If TRUE compute transition rates using weights specified in seqdata.
transition	Only used if time.varying=TRUE. If transition="both", it uses the transition rates from previous and next state. It can also be set to "previous" or "next".
lag	Integer. Only used with (method="TRATE"). Time between the two states considered to compute transition rates (one by default).
missing.trate	Logical. Only used with (method="TRATE"). If TRUE, substitution costs with missing state are also based on transition rates. If FALSE (default value), the substitution cost for the missing state are set to miss.cost.

Details

The substitution-cost matrix has dimension $ns * ns$, where ns is the number of states in the [alphabet](#) of the sequence object. The element (i, j) of the matrix is the cost of substituting state i with state j.

With the "CONSTANT" method, the substitution costs are the same for all the states, with a default value of 2. An alternative value can be provided by the user. When the "TRATE" (transition rates) method is chosen, the transition rates between all states are computed using the [seqtrate](#) function. The substitution cost between states i and j is obtained with the formula

$$SC(i, j) = cval - P(i, j) - P(j, i)$$

where $P(i, j)$ is the transition rate from state i to j.

Author(s)

Matthias Studer and Alexis Gabadinho (first version) (with Gilbert Ritschard for the help page)

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2010). Mining Sequence Data in R with the TraMineR package: A user's guide. Department of Econometrics and Laboratory of Demography, University of Geneva.

See Also

[seqtrate](#), [seqdef](#), [seqdist](#).

Examples

```
## Defining a sequence object with columns 10 to 25
## in the 'biofam' example data set
data(biofam)
biofam.seq <- seqdef(biofam,10:25)

## Optimal matching using transition rates based substitution-cost matrix
## and insertion/deletion costs of 3
trcost <- seqsubm(biofam.seq, method="TRATE")
biofam.om <- seqdist(biofam.seq,method="OM",indel=3,sm=trcost)

## Optimal matching using constant value (2) substitution-cost matrix
## and insertion/deletion costs of 3
ccost <- seqsubm(biofam.seq, method="CONSTANT", cval=2)
biofam.om.c2 <- seqdist(biofam.seq, method="OM",indel=3,sm=ccost)

## Displaying the distance matrix for the first 10 sequences
biofam.om.c2[1:10,1:10]

## ================================
## Example with weights and missings
## ================================
data(ex1)
ex1.seq <- seqdef(ex1,1:13, weights=ex1$weights)

## Unweighted
subm <- seqsubm(ex1.seq, method="TRATE", with.missing=TRUE, weighted=FALSE)
ex1.om <- seqdist(ex1.seq, method="OM", sm=subm, with.missing=TRUE)

## Weighted
subm.w <- seqsubm(ex1.seq, method="TRATE", with.missing=TRUE, weighted=TRUE)
ex1.omw <- seqdist(ex1.seq, method="OM", sm=subm.w, with.missing=TRUE)

ex1.om == ex1.omw
```

---

seqsubsn   *Number of distinct subsequences in a sequence.*

Description

Computes the number of distinct subsequences in a sequence using Elzinga's algorithm.

Usage

```
seqsubsn(seqdata, DSS=TRUE)
```

Arguments

seqdata   a state sequence object as defined by the [seqdef](#) function.

DSS   if TRUE, the sequences of Distinct Successive States (DSS, see [seqdss](#)) are first extracted (e.g., the DSS contained in `'D-D-D-D-A-A-A-A-A-A-A-D'` is `'D-A-D'`), and the number of distinct subsequences in the DSS is computed. If FALSE, the number of distinct subsequences is computed from sequences as they appear in the input sequence object. Hence the number of distinct subsequences is in most cases much higher with the DSS=FALSE option.

Details

The function first searches for missing states in the sequences and if found, adds the missing state to the alphabet for the extraction of the distinct subsequences. A missing state in a sequence is considered as the occurrence of an additional symbol of the alphabet, and two or more consecutive missing states are considered as two or more occurrences of the same state. The with.missing=TRUE argument is used for calling the [seqdss](#) function when DSS=TRUE.

Value

Vector with the number of distinct subsequences for each sequence in the input state sequence object.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

See Also

[seqdss](#).

Examples

```
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## Number of subsequences with DSS=TRUE
seqsubsn(actcal.seq[1:10,])

## Number of subsequences with DSS=FALSE
seqsubsn(actcal.seq[1:10,],DSS=FALSE)
```

---

seqtab   *Frequency table of the sequences*

Description

Computes the frequency table of the sequences (count and percent of each sequence).

Usage

```
seqtab(seqdata, tlim=1:10, weighted=TRUE, format="SPS")
```

Arguments

seqdata	a sequence object as defined by the [seqdef](#) function.
tlim	returns the table for the sequences at ranks 'tlim' in the list of distinct sequences sorted in decreasing order of their frequencies. Default is 1:10, i.e. the 10 most frequent sequences. Can be any subset, like 5:10 (fifth to tenth most frequent sequences) or c(2,10) (second and tenth most frequent sequences). Set tlim=0 to get the table for the whole set of distinct sequences.
weighted	if TRUE (default), frequencies account for the weights, if any, assigned to the state sequence object (see [seqdef](#)). Set to FALSE for ignoring weights.
format	format used for displaying the rownames (the sequences) in the output table. Default is SPS format, which yields shorter and more readable sequence representations. Alternatively, "STS" may be specified.

Details

The weighted argument has no effect when no weights were assigned to the state sequence object since weights default in that case to 1.

Value

An object of class stslist.freq. This is actually a state sequence object (containing a list of state sequences) with added attributes, among others the freq attribute containing the frequency table. There are print and plot methods for such objects. More sophisticated plots can be produced with the seqplot function.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

See Also

seqplot, plot.stslist.freq.

Examples

```
## Creating a sequence object from the actcal data set
data(actcal)
actcal.lab <- c(”> 37 hours”, ”19-36 hours”, ”1-18 hours”, ”no work”)
actcal.seq <- seqdef(actcal, 13:24, labels=actcal.lab)

## 10 most frequent sequences in the data
seqtab(actcal.seq)

## With tlim=0, we get all distinct sequences in the data set
## sorted in decreasing order of their frequency
seqtab(actcal.seq, tlim=0)

## Example with weights
## from biofam data set using weigths
data(ex1)
ex1.seq <-  seqdef(ex1, 1:13, weights=ex1$weights)

## Unweighted frequencies
seqtab(ex1.seq, weighted=FALSE)

## Weighted frequencies
seqtab(ex1.seq, weighted=TRUE)
```

---

seqtransn   *Number of transitions in a sequence*

Description

Computes the number of transitions in each sequence of a sequence object.

Usage

```
seqtransn(seqdata, with.missing=FALSE, norm=FALSE, pweight=FALSE)
```

Arguments

seqdata	a state sequence object as defined by the seqdef function.
with.missing	logical. if set as TRUE, missing states (gaps in sequences) are considered as an additional state and included in the DSS sequence. See seqdss.
norm	logical. If set as TRUE, the number of transitions is divided by its theoretical maximum, the length of the sequence minus 1. When length of the sequence is 1, normalized value is set to 0 as in the non-normalized case.
pweight	logical. EXPERIMENTAL! If set as TRUE, when counting transitions each transition does not account for 1 but for its probability (transition rate) as observed in the data.

Details

A transition in a sequence is a state change between time/position t and $t + 1$. For example, the sequence "A-A-A-A-B-B-A-D-D-D" contains 3 transitions. The maximum number of transitions a sequence can contain is $\ell - 1$ where ℓ is the length of the sequence. The number of transitions is obtained by subtracting 1 to the length of the the Distinct Successive State (DSS) sequence.

Value

a state sequence object containing the number of transitions of each sequence in the object given as argument.

Author(s)

Alexis Gabadinho (with Gilbert Ritschard for the help page)

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

See Also

seqdss.

Examples

```
## Creating a sequence object from columns 13 to 24
## in the 'actcal' example data set
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## Computing the number of transitions
actcal.trans <- seqtransn(actcal.seq)

## Displaying the DSS for the first 10 sequences
actcal.trans[1:10]

## Example with with.missing argument
data(ex1)
ex1.seq <- seqdef(ex1, 1:13)

seqtransn(ex1.seq)
seqtransn(ex1.seq, with.missing=TRUE)
```

| seqtrate | *Compute transition rates between states* |

Description

Returns a matrix with transition rates between states, computed from a set of sequences.

Usage

```
seqtrate(seqdata, statl=NULL, time.varying=FALSE, weighted=TRUE,
        lag=1, with.missing=FALSE)
```

Arguments

seqdata	a sequence object as defined by the [seqdef](#) function.
statl	a list of states or events for which the transition rates will be computed. If omitted (default), transition rates are computed between the distinct states in seqdata (obtained with the [alphabet](#) function).
time.varying	Logical. If TRUE, return an [array](#) containing a distinct matrix for each time unit. The time is the third dimension (subscript).
weighted	Logical. If TRUE, compute transition rates using weights specified in seqdata.
lag	Integer. Time between the two states considered to compute transition rates (one by default).
with.missing	Logical. If FALSE (default value), returned transition rates ignore missing values.

Details

Transition rates are the probabilities of transition from one state to another observed in the sequence data. Substitution costs based on transition rates can be used when computing distances between sequences with the optimal matching method (see [seqdist](#)).

Value

a matrix of dimension $ns * ns$, where ns is the number of states in the [alphabet](#) of the sequence object.

Author(s)

Matthias Studer and Alexis Gabadinho (first version) (with Gilbert Ritschard for the help page)

References

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. *Journal of Statistical Software* **40**(4), 1-37.

See Also

[seqdist](#) [seqsubm](#) [alphabet](#).

Examples

```
   ## Loading the 'actcal' example data set
   data(actcal)

   ## Defining a sequence object with data in columns 13 to 24
   ## (activity status from january to december 2000)
   actcal.seq <- seqdef(actcal,13:24,informat='STS')

   ## Computing transition rates
   seqtrate(actcal.seq)

   ## Computing transition rates between states "A" and "B" only
   seqtrate(actcal.seq, c("A","B"))

## ====================
## Example with weights
## ====================
data(ex1)
ex1.seq <- seqdef(ex1,1:13, weights=ex1$weights)

seqtrate(ex1.seq, weighted=FALSE)
seqtrate(ex1.seq, weighted=TRUE)
```

---

seqtree   *Tree structured analysis of a state sequence object.*

Description

Facility for growing a regression tree for a state sequence object.

Usage

```
seqtree(formula, data = NULL, weighted = TRUE, minSize = 0.05,
    maxdepth = 5, R = 1000, pval = 0.01,
    weight.permutation = "replicate",
    seqdist_arg = list(method = "LCS", norm = TRUE),
    diss = NULL, squared = FALSE, first = NULL)
```

Arguments

formula	a formula where the left hand side is a state sequence object (see [seqdef](#)) and the right hand specifies the candidate variables for partitioning the set of sequences.
weighted	Logical. If TRUE, use the weights of the state sequence object.
data	a data frame where variables in the formula will be searched

minSize	minimum number of cases in a node, in percentage if less than 1.
maxdepth	maximum depth of the tree.
R	Number of permutations used to assess the significance of the split.
pval	Maximum p-value, in percent.
weight.permutation	
	Weights permutation method: "diss" (attach weights to the dissimilarity matrix), "replicate" (replicate case according to the weights arguments), "rounded-replicate" (replicate case according to the rounded weights arguments), "random-sampling" (random assignment of covariate profiles to the objects using distributions defined by the weights.)
seqdist_arg	list of arguments directly passed to [seqdist](), only used if diss=NULL
diss	An optional dissimilarity matrix. If not provided, a dissimilarity matrix is computed using [seqdist]() and seqdist_arg
squared	Logical. If TRUE, the dissimilarity matrix is squared
first	Character. An optional variable name to force the first split.

Details

The function provides a simplified interface for applying [disstree]() on state sequence objects.

The seqtree objects can be "plotted" with [seqtreedisplay](). A print method is also available which prints the medoid sequence for each terminal node.

Value

A seqtree object with same attributes as [disstree]() objects.

The leaf membership is in the first column of the fitted attribute. For example, the leaf memberships for a tree dt are in dt$fitted[,1].

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

References

Studer, M., G. Ritschard, A. Gabadinho and N. S. Müller (2011). Discrepancy analysis of state sequences, *Sociological Methods and Research*, Vol. 40(3), 471-510.

See Also

[seqtreedisplay](), [disstree]()

Examples

```
data(mvad)

## Defining a state sequence object
mvad.seq <- seqdef(mvad[, 17:86])

## Growing a seqtree from Hamming distances:
##   Warning: The R=10 used here to save computation time is
##   much too small and will generate strongly unstable results.
##   We recommend to set R at least as R=1000.
seqt <- seqtree(mvad.seq~ male + Grammar + funemp + gcse5eq + fmpr + livboth,
    data=mvad, R=10, seqdist_arg=list(method="HAM", norm=TRUE))
print(seqt)

## Growing a seqtree from an existing distance matrix
mvad.dhd <- seqdist(mvad.seq, method="DHD")
seqt <- seqtree(mvad.seq~ male + Grammar + funemp + gcse5eq + fmpr + livboth,
    data=mvad, R = 10, diss=mvad.dhd)
print(seqt)


### Following commands only work if GraphViz is properly installed
## Not run:
seqtreedisplay(seqt, type="d", border=NA)
seqtreedisplay(seqt, type="I", sortv=cmdscale(mvad.dhd, k=1))

## End(Not run)
```

---

seqtreedisplay	*Graphical rendering of a sequence regression tree*

Description

Generate a graphical representation of a regression tree of state sequence data.

Usage

```
seqtreedisplay(tree, filename=NULL, seqdata=tree$info$object,
                imgLeafOnly=FALSE, sortv=NULL, dist.matrix=NULL,
                title.cex=3, withlegend="auto", legend.fontsize=title.cex,
                axes=FALSE, imageformat="png", withquality=TRUE,
                quality.fontsize=title.cex, legendtext=NULL,
                showtree=TRUE, showdepth=FALSE, ...)

disstreedisplay(tree, filename=NULL, imagedata=NULL, imagefunc=plot,
                imgLeafOnly=FALSE, title.cex=3, imageformat="png",
                withquality=TRUE, quality.fontsize=title.cex,
                legendtext=NULL, showtree=TRUE, showdepth=FALSE, ...)
```

Arguments

tree	A seqtree object (as produced by [seqtree](#)) for seqtreedisplay. A disstree object (as produced by [disstree](#)) for disstreedisplay.
filename	The name of a file where to save the plot (overwritting existing file). If NULL, a temporary file is created.
seqdata	The sequence object containing the state sequences plotted in the nodes.
imgLeafOnly	Logical. If TRUE sequences are plotted only in terminal nodes.
sortv	Argument passed to [seqplot](#)
dist.matrix	Argument passed to [seqplot](#)
title.cex	The cex value for the node titles (see [par](#)).
withlegend	Logical. Should the color legend be displayed on the plot?
legend.fontsize	
	Font cex value for the legend.
axes	Argument passed to [seqplot](#)
imageformat	Image format of the output file (filename)
withquality	If TRUE, a node displaying fitting measures of the tree is added to the plot.
quality.fontsize	
	Numeric. Size of the font of the fitting measures node.
legendtext	Character. Optional text information that should be added.
showtree	Logical. Should the tree be shown on the screen?
showdepth	Logical. If TRUE, the splits are ordered according to their global pseudo-R2.
imagefunc	A function to plot the individuals in a node, see details.
imagedata	a data.frame that will be passed to imagefunc.
...	additional arguments passed to seqplot

Details

This function generates a tree image. For each node, it invokes [seqplot](#) for the selected lines of seqdata as argument. You should at least specify the type of the plot to use (type="d" for instance, see [seqplot](#) for more details).

The plot is actually not generated as an R plot, but with GraphViz (www.graphviz.org). Hence, seqtreedisplay only works when GraphViz is correctly installed.

Conversion to image formats other than "jpeg" or "png" is done using ImageMagick (www.imagemagick.org). To use this feature, ImageMagick (www.imagemagick.org) should hence also be installed.

Value

None

Author(s)

Matthias Studer (with Gilbert Ritschard for the help page)

See Also

See [seqtree](#) and [disstree](#) for examples, and [disstree2dot](#) for generating "dot" files.

---

stlab   *Get or set the state labels of a sequence object*

Description

This function gets or sets the state labels of a sequence object, that is, the long labels used when displaying the state legend in plotting functions.

Usage

```
stlab(seqdata)
stlab(seqdata) <- value
```

Arguments

seqdata   a state sequence object as defined by the [seqdef](#) function.

value   a vector of character strings containing the labels, of length equal to the number of states in the alphabet. Each string is attributed to the corresponding state in the alphabet, the order being the one returned by the [alphabet](#).

Details

The state legend is plotted either automatically by the plot functions provided for visualizing sequence objects or with the [seqlegend](#) function. A long label is associated to each state of the alphabet and displayed in the legend. The state labels are defined when creating the sequence object, either automatically using the values found in the data or by specifying a user defined vector of labels. The stlab function can be used to get or set the state labels of a previously defined sequence object.

Value

For 'stlab' a vector containing the labels.

For 'stlab<-' the updated sequence object.

See Also

[seqdef](#)

Examples

```
## Creating a sequence object with the columns 13 to 24
## in the 'actcal' example data set
## The color palette is automatically set
data(actcal)
actcal.seq <- seqdef(actcal,13:24)

## Retrieving the color palette
stlab(actcal.seq)
seqiplot(actcal.seq)

## Changing the state labels
stlab(actcal.seq) <- c("Full time","Part time (19-36 hours)",
    "Part time (1-18 hours)", "No work")
seqiplot(actcal.seq)
```

TraMineR.checkupdates   *Check for TraMineR updates*

Description

Check if the installed version of TraMineR is up-to-date. This function only prints a message and does not need any argument. It connects to the TraMineR webserver (http://mephisto.unige.ch/traminer).

Usage

```
TraMineR.checkupdates()
```

Value

Return your current version number of TraMineR and the latest stable and development version number if more recent versions are available.

Author(s)

Nicolas S. Müller

TraMineRInternal   *Access to TraMineR internal functions*

Description

Functions allowing other packages to access some TraMineR internal functions. Corresponding functions are respectively TraMineR.setlayout, TraMineR.Legend, DTNInit, seqeage, seqgbar and DTNsplit. For experts only.

Usage

```
TraMineRInternalLayout(...)
TraMineRInternalLegend(...)
TraMineRInternalNodeInit(...)
TraMineRInternalSeqeage(...)
TraMineRInternalSeqgbar(...)
TraMineRInternalSplitInit(...)
```

Arguments

...	Arguments passed to or from other methods.

Index