Package 'TraMineRextras'

October 29, 2013

Version 0.2.2

Date 2013-10-30

Title Extras for use with the TraMineR package

Depends R (>= 2.8.1), TraMineR (>= 1.8-4), RColorBrewer, combinat,survival

Suggests cluster

Description
Collection of ancillary functions and utilities to be used in conjunction with the TraMineR package for sequence data exploration. Most of the functions are in test phase, lack systematic consistency check of the arguments and are subject to changes. Once fully checked, some of the functions of this collection could be included in a next release of TraMineR.

License GPL (>= 2)

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

Encoding UTF-8

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

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-10-29 18:04:08

R topics documented:

`convert` *Converting between graphical formats*

Description

Wrapper function for converting graphics with ImageMagick

Usage

```
convert.g(path = NULL, fileroot= "*", from = "pdf",
    to = "png", create.path = TRUE, options = NULL)
```

Arguments

`path`	String: The path to the from graphic files.
`fileroot`	String: Graphic root name; default is "*" for all files with the `from` extension.
`from`	File type extension specifying the from format.
`to`	File type extension specifying the to format.
`create.path`	Logical: Should the output files be placed in a `to` subfolder.
`options`	Additional options to be passed to the ImageMagick `mogrify` function

Details

Conversion is done through a call to ImageMagick `mogrify` function. This means that ImageMagick should be installed on your system. It must also be listed in the path.

for some values such as `"pdf"` and `"eps"` of the `from` or `to` arguments ImageMagick works in conjunction with Gostscript. The latter should, therefore, also be accessible.

See Also

png, pdf

Examples

```
## Not run:
## Convert all .pdf graphics in the "figSW" directory
## into .png files and put the files in a "png" subfolder.
convert.g(path="figSW", from="pdf", to="png")

## Same, but convert to .jpg files.
convert.g(path="figSW", to="jpg")

## convert file "example.eps" in current path to ".pdf"
## and put it in same folder.
convert.g(fileroot = "example", create.folder=FALSE)

## End(Not run)
```

---

createdatadiscrete *Transform time to event data into a discrete data format*

Description

Transform time to event data (in a specific format, see the detailed description below) into a person-period data format suitable for automatic sequential association rules extraction

Usage

```
createdatadiscrete(ids, data, vars, agemin, agemax,
    supvar=NULL)
```

Arguments

`ids`	a vector containing an unique identification number for each case
`data`	a data frame containing time to event data, with variables containing the durations called as in the vars argument, and those with the censoring indicators named as in the vars argument followed by "ST" (for example column A is duration until event A, and column AST is the censoring indicator). This data frame must contain an unique identification variable named "IDPERS".

vars	a vector with the names of the duration variables
agemin	a data frame with two variables : "IDPERS" for the unique identification variable, and "AGE" for the starting time of the observation
agemax	a data frame with two variables : "IDPERS" for the unique identification variable, and "AGE" for the ending time of the observation
supvar	a vector of variables to add to the resulting person-period data frame

Details

The data frame from the data argument must contain two variables for each event: a duration variable that indicates the time when the event occured, and a status variable that indicates if the event occurred (1) or not (0). If the event did not occur, the observation for this individual will go until the age specified through the agemax argument. Each status variable must have the same name than the duration variable, followed by "ST". For example, if the duration variable for an event "divorce" is called "div", then the status variable has to be named "divST".

The result from this function is a list with one person-period data frame by event, where the dependent event is different each time. Please see the attached data file and code for an example.

The resulting object is one of the required argument for the seqerulesdisc function that computes the association rules, the hazard ratios and the p-values, using discrete-time regressions. Unlike the method presented in Müller et al. 2010, this function does not use Cox proportional hazard models, but discrete-time regression models with a complementary log-log link function, which gives similar results.

Value

a list with one person-period data frame by event, where the dependent event is different each time. Please see the attached data file and code for an example.

Author(s)

Nicolas S. Müller

References

Müller, N.S., M. Studer, G. Ritschard et A. Gabadinho (2010), Extraction de règles d'association séquentielle à l'aide de modèles semi-paramétriques à risques proportionnels, *Revue des Nouvelles Technologies de l'Information*, **Vol. E-19**, EGC 2010, pp. 25-36

See Also

seqerulesdisc to compute the association rules.

Examples

```
##
```

ctplot *Plot individual trajectories of longitudinal categorical data.*

Description

Illustrates individual categorical trajectories in a modified time-series plot. The x scale represents the order position (often the timestamp), the y the response category (events or state).

Usage

```
ctplot(x,y,id,weights=NULL,
       cov=NULL,subset=NULL,type="distinctive",
       embedding="most-frequent",x.order=FALSE,
       x.orderalign=ifelse(split=="last","last","first"),
       y.optimalphabet=FALSE,R=NULL,
       main="",sub=NULL,mtext=TRUE,
       xlab="order position",ylab="",xlim=NULL,ylim=NULL,
       grid.col="white",grid.fill="grey90",
       grid.scale=1/4,grid.shape="default",grid.lwd=0,
       cpal=NULL,alpha=1,lcourse="upwards",lorder="background",
       lweights=TRUE,lwd.min=0.5,lwd.max=2,lty=1,pch=4,
       cex=1,border=grid.col,border.lwd=grid.lwd/2,
       sf.cex=1,sf.cex.leaves=1,sf.pch=16,rotate=FALSE,
       split=NULL,layout=NULL,
       show.type=1,show=c(0,1),hide.col="grey75",
       hide.s=0.1,hide.na.cost=1,
       print=FALSE,return.data=FALSE,...)
```

Arguments

x	The order position vector.
y	The response vector (i.e events, ordinal responses etc.).
id	The subject identification vector. Either a factor or an integer vector.
weights	A weights vector that corresponds to the subjects defined by the id vector. If id is a factor, the length of weights must be equal to the number of levels of the id vector. The first value of weights is the assigned to the first id label etc.. If id is an integer vector, the length of weights must be equal to the number of distinctive numbers in id.
cov	A grouping vector which splits the plot into one panel per group. This vector must be defined analogous as the "weights" vector.
subset	An optional vector of id labels which should be omitted.
type	The type of trajectories to draw. Either "distinctive" or "non-embeddable".

embedding	Option for type="non-embeddable", the method how to assign trajectories having multiple corresponding non-embeddable trajectories. Either "most-frequent" (default) or "uniformly".
x.order	Logical. Transforms the x vector into individual integer orders
x.orderalign	Alignment mode for data where the order positions are individual integer orders. align="first" aligns the trajectories left hand, align="last" right hand. Assigning an y category produces an alinement of the first occurrences of this category.
y.optimalphabet	Logical. Indicates if arrangements of y categories should be optimized. Works only with a limited number of categories.
R	Accelerate the y order optimisation procedure. Set a number in case of y provides many categories.
main	A main title for the plot, see also "title".
sub	Subtitles. Used in case of multiple plot panels.
mtext	Logical. Print panel information or not.
xlab	A label for the x axis, defaults to a description of "x".
ylab	A label for the y axis, defaults to a description of "y".
xlim	The x limits (x1, x2) of the plot.
ylim	The y limits (y1, y2) of the plot.
grid.col	Color of border of underlaid rectangles.
grid.fill	Color of underlaid rectangles.
grid.scale	Expansion degree of underlaid rectangles.
grid.shape	Either "default" or "proportional".
grid.lwd	Line width or border of underlaid rectangles.
cpal	A colour palette to be assigned to the sequences.
alpha	Degree of line and symbol transparency. Choose a number between 0 and 1.
lorder	Either "background" (default) or "foreground". The first plots infrequent trajectories, the latter the frequent trajectories in the front.
lcourse	Handling for line connection of simultaneous observations. Either "upwards" or "downwards".
lweights	logical: Should the line width be proportional to the represented trajectory? If FALSE, line width is set as lwd.min.
lwd.min	The minimal line width to be drawn in the plot.
lwd.max	The maximal line width to be drawn in the plot.
lty	Line type of lines connecting succeeding observations.
pch	"pch" the plotting symbols: see "points". Used if lweights=FALSE.
cex	Expansion factor for the plotted squared symbols.
border	Color of symbol borders.
border.lwd	Line widths of symbol borders
sf.cex	Expansion factor of the center points of plotted sunflowers.

sf.cex.leaves	Expansion factor for leaves the of plotted sunflowers.
sf.pch	Plotting character to be used for points (number[i]==1) and center of sunflowers.
rotate	Logical. If TRUE, plotted sunflowers are rotated randomly.
split	Logical value for plot panel arrangement modes. If TRUE and x.orderalign="first", the plot produces one plot panel per observed initial event and inscribes the groups which are initialised by the according event. The case TRUE and align="last" proceeds analogously but considering final events.
layout	Integer vector of length 2. Determines the number of rows and columns of the plot panels arrangement.
show.type	Criteria which trajectories to set in the foreground. Type 1 (default) shows trajectories which present relative frequencies between the two values assigned to the "show" argument. Type 2 shows segments of trajectories.
show	Vector of two values between 0 and 1. Indicate the minimal and maximal relative frequency for a trajectory to be presented in the foreground of the plot.
hide.col	color of trimmed trajectories. If set to "white" trajectories are not shown at all.
hide.s	Option for ordinal sequences. Experimental.
hide.na.cost	Option of for ordinal sequences. Experimental.
print	Prints the current step of the plot procedure.
return.data	Returns the plot data in a list.
...	Arguments to be passed to methods, such as graphical parameters (see par).

Details

Coming soon.

References

Bürgin, R., G. Ritschard and E. Rousseaux (2012), "Exploration graphique de données séquentielles". In Atelier Fouille Visuelle de Données : méthologie et évaluation, EGC 2012, Bordeaux. Association EGC: http://www.egc.asso.fr/sdoc-237-egc2012_atelier_fvd.pdf .

Examples

```
## ==================
## the plot principle
## ==================

x <- c(1,2,2,2,3,4,1,2,2,2,3,4,1,1.5,2,2,3,3.5,4,4,1,2,2,2,3,4,
       1,2,3,4,3.5,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,1,2,
       1,2,3,4,1,2)

y <- factor(c("A","B","B","B","C","A","A","B","B","B","C","A","A",
              "B","B","C","C","C","C","C","A","B","B","B","C","A",
              "A","A","A","A","A","B","A","A","A","A","A","A","A",
```

```
              "A","A","A","A","A","A","A","A","A","A","A","A","A",
              "A","A","A","A","A","B","C","A","A","A"))

id <- factor(c(1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3,3,3,4,4,4,4,4,4,4,
               5,5,5,5,6,6,7,7,7,7,8,8,8,8,9,9,9,9,10,10,10,10,
               11,11,11,11,12,12,13,13,14,14,14,14,15,15))

ctplot(x,y,id,lwd.min=5,lwd.max=12,cex=1,
                  type="non-embeddable",alpha=0.9,
                  sf.cex=0.5,sf.cex.leaves=1,grid.scale=0.4)


## ====================
## plot the biofam data
## ====================

## loading the data and defining an event sequence dataset
## ========================================================

data(biofam)
lab <- c("Parent","Left","Married",
         "Left+Marr","Child","Left+Child",
         "Left+Marr+Child","Divorced")
biofam.seq <- seqdef(data=biofam,var=10:25,labels=lab)
biofam.TSE <- seqformat(data=biofam.seq,from="STS",to="TSE",
                        tevent=seqetm(seq=biofam.seq,method="state"))
biofam.TSE$event <- factor(biofam.TSE$event,levels=lab)

## plot the data
## =============

par(mar=c(4,8,2,2))
ctplot(x=biofam.TSE$time,
                  y=biofam.TSE$event,
                  id=biofam.TSE$id,x.order=TRUE,
                  type="non-embeddable",lwd.max=10)

par(mar=c(4,8,2,2))
ctplot(x=biofam.TSE$time,y=biofam.TSE$event,
                  id=biofam.TSE$id,split="first",layout=c(2,1),
                  grid.scale=0.5,x.order=TRUE)

par(mar=c(4,8,2,2))
ctplot(x=biofam.TSE$time,y=biofam.TSE$event,
                  id=biofam.TSE$id,
                  x.order=TRUE,x.orderalign="last",split="last",
                  layout=c(4,2),lwd.max=2.5,grid.scale=0.5)
```

---

dissvar.grp *Discrepancy by group.*

Description

This function computes the dissimilarity-based discrepancy measure of the groups defined by the group variable. The function is a wrapper for the TraMineR dissvar function.

Usage

```
dissvar.grp(mdis, group=NULL, ...)
```

Arguments

mdis	a dissimilarity matrix or a dist object.
group	group variable. If NULL a single group is assumed.
...	additional arguments passed to dissvar .

Details

The function is a wrapper for running dissvar on the different groups defined by the group variable.

Value

A vector with the group discrepancy measures.

Note

This function is a pre-release and further testing is still needed, please report any problems.

Author(s)

Gilbert Ritschard

See Also

dissvar

Examples

```
## create the biofam.seq state sequence object from the biofam data.
data(biofam)
biofam <- biofam[1:100,]
biofam.seq <- seqdef(biofam[,10:25])
dist <- seqdist(biofam.seq, method="HAM")

## discrepancy based on non-squared dissimilarities
dissvar.grp(dist, biofam$plingu02)
## square root of discrepancy based on squared dissimilarities
sqrt(dissvar.grp(dist, biofam$plingu02, squared=TRUE))
```

FCE_to_TSE *Data conversion from Fixed Column Event format to TSE.*

Description

Data conversion from Fixed Column Event format to TSE.

Usage

```
FCE_to_TSE(seqdata, id = NULL, cols, eventlist = NULL, firstEvent = NULL)
```

Arguments

seqdata	data frame or matrix containing event sequence data in FCE format.
id	column containing the identification numbers for the sequences.
cols	Real. Column containing the timing of the event. A missing value is interpreted as a non-occurrence of the event.
eventlist	Event names, specified in the same order as cols argument. If NULL (default), column names are used.
firstEvent	Character. The name of an event to be added at the beginning of each event sequences. This allows to include individuals with no events. If NULL (default), no event is added.

Details

The usual data format for event sequence is TSE (see seqecreate).

Value

A data.frame with three columns: "id", "timestamp" and "event".

Note

This function is a pre-release and further testing is still needed, please report any problems.

Author(s)

Matthias Studer

See Also

seqecreate, seqformat

Examples

```
## Generate a ramdom data set
fce <- data.frame(id=1:100, event1=runif(100), event2=runif(100))

## Add missing values (ie non-occurrences)
fce[runif(100)<0.1, "event1"] <- NA
fce[runif(100)<0.1, "event2"] <- NA

tse <- FCE_to_TSE(fce, id="id", cols=c("event1", "event2"),
       eventlist=c("Marriage", "Child birth"), firstEvent="Birth")

seq <- seqecreate(tse)
print(seq[1:10])
```

`group.p` *Adds proportion of occurrences to each level names*

Description

Adds the proportion of occurrences of each level to the corresponding level name.

Usage

```
group.p(group, weights=NULL)
```

Arguments

`group`	A group variable.
`weights`	Vector of weights of same length as the group variable.

Details

The `group` variable can be a factor or a numerical variable. In the latter case it is transformed to a factor.

Author(s)

Gilbert Ritschard

See Also

`seqplot`.

Examples

```
data(actcal)
actcal <- actcal[1:100,]
actcal.seq <- seqdef(actcal[,13:24])
seqdplot(actcal.seq, group=group.p(actcal$sex))

levels(group.p(actcal$sex, weights=runif(length(actcal$sex))))
```

HSPELL_to_STS *Data conversion from Horizontal Spell to STS.*

Description

Convert data from Horizontal Spell to STS.

Usage

```
HSPELL_to_STS(seqdata, begin, end, status = NULL,
    fixed.status = NULL, pvar = NULL, overwrite = TRUE,
    fillblanks = NULL, tmin = NULL, tmax = NULL, id = NULL,
    endObs = NULL)
```

Arguments

seqdata	a data frame or matrix containing sequence data.
begin	Vector containing the columns (name or number) with the beginning position of each spell.
end	Vector containing the columns (name or number) with the end position of each spell.
status	Vector containing the columns (name or number) with the status of each spell.
fixed.status	Default status (for period not covered by any spell.)
pvar	names or numbers of the column containing the 'birth' time.
overwrite	Should the most recent episode overwrite the older one when they overlap? If FALSE, the most recent episode starts from the end of the previous one.
fillblanks	If not NULL, character used for filling gaps between episodes.
tmin	If sequences are to be defined on a calendar time axis, it defines the starting time of the axis. If set as NULL, the start time is set as the minimum of the 'begin' column in the data.
tmax	If year sequences are wanted, defines the ending year of the sequences. If set to NULL, it is guessed from the data (not so accurately!).
id	column containing the identification numbers for the sequences.
endObs	An optional end of observation date. Usefull for retrospective survey.

Details

Hortizontal spell data format has the following caracteristics: - One row per individual - Each spell is specified with three consecutive variables: a begin date, an end date, and the status. - For unused spells, begin and end values should be set as NA.

Value

A data.frame with the sequence in STS format.

Note

This function is a pre-release and further testing is still needed, please report any problems.

Author(s)

Matthias Studer

See Also

See Also seqformat.

Examples

```
hspell <- data.frame(begin1=rep(1, 5), end1=c(2:5, NA), status1=1:5,
                     begin2=c(3:6, NA), end2=rep(NA, 5), status2=5:1)
sts <- HSPELL_to_STS(hspell, begin=c("begin1", "begin2"), end=c("end1", "end2"),
                     status=c("status1", "status2"))
```

---

pamward *PAM from k-solution of hierarchical clustering*

Description

Runs a pam clustering (pam) from the solution in k groups of a hierarchical clustering (agnes) .

Usage

```
pamward(dist, k=3, method="ward")
```

Arguments

dist	A distance matrix or object.
k	Number of clusters.
method	Method for the hierarchical clustering (see agnes)

Details

The function first runs the hierarchical clustering, retrieves the medoids of the solution for the provided k and uses those medoids as start centers for the pam partitioning.

Value

An object of class "pam". See pam.object for details.

Author(s)

Gilbert Ritschard

See Also

agnes and pam.

Examples

```
library(cluster)
data(actcal)
actcal.seq <- seqdef(actcal[1:200,13:24])
actcal.ham <- seqdist(actcal.seq, method = "HAM")
clust <- pamward(actcal.ham, k = 4)
table(clust$clustering)
```

---

plot.emlt *Emlt Plotting*

Description

Plotting, resulting from emlt, static and dynamic states structure in a sequence analysis - Two types of plot : The evolve in time of correlation between states and the projection of states/situations on their principal planes

Usage

```
## S3 method for class 'emlt'
plot(x, from, to, delay=NULL, leg=TRUE, type="cor", cex=0.7, compx=1, compy=2, ...)
```

Arguments

x	a sequence analysis producted by emlt
type	what type of plot should be drawn. Possible types are "cor" for the evolve in time of correlation between states "pca" for the projection of states/situations on their principal planes
from	for type "cor" first states to compare, may be a vector
to	for type "cor" second states to compare

delay	for type "cor", the delay or step between "from" and "to" arguments. The correlation between state "from" at time t and "to" at t+delay. By default delay is 0.
compx	for type "pca" first component, axis x
compy	for type "pca" second component, axis y
leg	A boolean argument TRUE, FALSE for including legend
cex	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default.
...	Arguments to be passed to methods, such as graphical parameters (see par)

Details

The evolve of the correlation reveals the evolve of the distance between situations/states index with time considering the emlt euclidean distance. The "pca" components are the components of the emlt transformed sequences, see seqemlt.

Author(s)

Patrick Rousset, Senior researcher at Cereq, rousset@cereq.fr with the help of Matthias Studer

References

- Rousset Patrick, Giret Jean-françois,Classifying Qualitative Time Series with SOM: The Typology of Career Paths in France Lecture Notes in computer science, vol 4507, 2007, Springer Berlin / Heidelberg - - Rousset Patrick, Giret Jean-françois, Yvette Grelet (2012) Typologies De Parcours et Dynamique Longitudinale, Bulletin de méthodologie sociologique, issue 114, april 2012. - - Rousset Patrick, Giret Jean-françois (2008) A longitudinal Analysis of Labour Market Data with SOM, Encyclopedia of Artificial Intelligence, Edition Information Science Reference -

See Also

See Also `seqemlt` (with examples)

Examples

```
## See examples on 'seqemlt' help page
```

---

rowmode *Modal state of a variable*

Description

Returns the modal state of a variable, e.g., the modal state in a sequence.

Usage

```
rowmode(v, except = NULL)
```

Arguments

v	A numerical or factor variable.
except	Vector of values that should be ignored; e.g., set except="*" to ignore missing states with default coding.

Details

The function tabulates the variable and returns the most frequent value.

Value

The modal value

Author(s)

Gilbert Ritschard

See Also

table.

Examples

```
data(actcal)
actcal.seq <- seqdef(actcal[1:10,13:24])
actcal.mod <- apply(as.matrix(actcal.seq), 1, rowmode)
head(actcal.mod)
```

---

seqauto *Auto-association between states*

Description

Computes auto-associations of order k = 1 to order, between current states and states lagged by k positions.

Usage

```
seqauto(seqdata, order = 1, measure = "cv")
```

Arguments

seqdata	A state sequence object or a data frame with sequential data in STS format.
order	Maximum wanted order of auto-association.
measure	Character string. Currently only "cv" (Cramer's v) is accepted.

Details

The function puts the data in "SRS" form by means of the seqformat function.

Value

A matrix with order rows and two columns: the auto-association and its p-value.

Warning

Function in development, not fully checked.

Author(s)

Gilbert Ritschard

See Also

seqformat

Examples

```
data(biofam)

biofam.seq <- seqdef(biofam[1:100,10:25])
aa <- seqauto(biofam.seq, order=5)
aa
```

---

seqe2stm *Definition of an events to states matrix.*

Description

This function creates a matrix specifying for each state (given in row) to which state we fall when the event given in column happens.

Usage

```
seqe2stm(events, dropMatrix = NULL, dropList = NULL, firstState = "None")
```

Arguments

events	Character. The vector of all possible events.
dropMatrix	Logical matrix. Specifying the events to forget once a given event has occurred.
dropList	List. Same as dropMatrix but using a list (often more convenient).
firstState	Character. Name of the first state, before any event has occurred.

Details

This function creates a matrix with in each cell the new state which results when the column event (column name) occurs while we are in the corresponding row state (row name). Such a matrix is required by TSE_to_STS. By default, a new state is created for each combination of events that already has occurred.

dropMatrix and dropList allow to specify which events should be "forgotten" once a given event has occurred. For instance, we may want to forget the "marriage" event once the event "divorce" has occurred.

dropMatrix specifies for each event given in row, the previous events, given in column that should be forgotten. dropList uses a list to specify the same things. The form is list(event1=c(..., events to forgets), event2=c(..., events to forgets)). See example below.

Value

A matrix.

Note

This function is a pre-release and further testing is still needed, please report any problems.

Author(s)

Matthias Studer

References

Ritschard, G., Gabadinho, A., Studer, M. & Müller, N.S. (2009), "Converting between various sequence representations", In Ras, Z. & Dardzinska, A. (eds) *Advances in Data Management*. Series: *Studies in Computational Intelligence*. Volume 223, pp. 155-175. Berlin: Springer.

See Also

TSE_to_STS

Examples

```
## Achieving same result using dropMatrix or dropList.
## List of possible events.
events <- c("marr", "child", "div")
dm <- matrix(FALSE, 3,3, dimnames=list(events, events))
dm[3, ] <- c(TRUE, TRUE, FALSE)
dm[1, 3] <- TRUE
## Using the matrix, we forget "marriage" and "child" events when "divorce" occurs.
## We also forget "divorce" after "marriage" occurs.
print(dm)
stm <- seqe2stm(events, dropMatrix=dm)

## Get same result with the dropList argument.
stmList <- seqe2stm(events, dropList=list("div"=c("marr", "child"), "marr"="div"))
```

```
## test that the results are the same
all.equal(stm, stmList)
```

`seqedist` *Distances between event sequences*

Description

Compute Optimal Matching like distances between event sequences. The distance measure is fully described in *Studer et al. 2010.*

Usage

```
seqedist(seqe, idcost, vparam, interval="No", norm="YujianBo")
```

Arguments

`seqe`	an event sequence object as defined by the `seqecreate` function.
`idcost`	Insertion/deletion cost of the different type of event (one entry per event type).
`vparam`	The cost of moving an event of one time unit.
`norm`	Character. One of `"YujianBo"` (respects triangle inequality), `"max"` (maximum distance) or `"none"`.
`interval`	Character. One of `"No"` (absolute ages), `"previous"` (time spent since previous event) or `"next"` (time spent until next event.).

Value

a distance matrix.

Author(s)

Matthias Studer

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.

Examples

```
data(actcal.tse)
actcal.seqe <- seqecreate(actcal.tse[1:200,])[1:6,]
## We have 8 different event in this dataset
idcost <- rep(1, 8)
dd <- seqedist(actcal.seqe, idcost=idcost, vparam=.1)
```

seqedplot *Graphical representation of a set of events sequences.*

Description

This function provides two ways to represent a set of events. The first one (type="survival") plots the survival curves of the first occurrence of each event. The second one (type="hazard") plots the mean counts of each events in a given time frame.

Usage

```
seqedplot(seqe, group = NULL, breaks = 20, ages = NULL, title = NULL,
   type = "survival", ignore = NULL, withlegend = "auto", cex.legend = 1,
   use.layout = (!is.null(group) | withlegend != FALSE),
   legend.prop = NA, rows = NA, cols = NA, axes = "all", xlab = "time",
  ylab = ifelse(type == "survival", "survival probability", "mean number of events"),
   cpal = NULL, ...)
```

Arguments

seqe	an event sequence object as defined by the seqecreate function.
group	Plots one plot for each level of the factor given as argument.
breaks	Number of breaks defining a period.
ages	Two numeric values representing minimum and maximum ages to be represented.
title	title for the graphic. Default is NULL.
type	the type of the plot. If type="survival", plots the survival curves of the first occurence of each event. If type="hazard", plots the mean numbers of each event in a given time frame.
ignore	Character. An optional list of events that will not be plotted.
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 values are "right" or FALSE. Obsolete value TRUE is equivalent to "auto".
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	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	optional arguments to arrange plots when use.layout=TRUE.

cols	optional arguments to arrange plots when use.layout=TRUE.
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.
xlab	an optional label for the x-axis. If set to NA, no label is drawn.
ylab	an optional label for the y-axis. If set to NA, no label is drawn.
cpal	Color palette used for the events. If NULL, a new color palette is generated.
...	Additionnal arguments passed to lines.

Author(s)

Matthias Studer

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.

Examples

```
data(actcal.tse)
actacl.tse <- actcal.tse[1:500,]
iseq <- unique(actcal.tse$id)
nseq <- length(iseq)
data(actcal)
actcal <- actcal[rownames(actcal) %in% iseq,]
actcal.seqe <- seqecreate(actcal.tse)
seqelength(actcal.seqe) <- rep(12, nseq)
seqedplot(actcal.seqe, type="hazard", breaks=6, group=actcal$sex, lwd=3)
seqedplot(actcal.seqe, type="survival", group=actcal$sex, lwd=3)
```

---

seqemlt *Euclidean Metric for Longitudinal Timelines*

Description

Computes a Euclidean distance between sequences. Transforms sequences such as distance beween sequences is equivalent to Euclidean distance between transformed sequences. The transformed sequences may be used as inputs of any Euclidean algorithm (clustering algorithms, ...). The distance is built considering the transitions between states at any step. A step weighing mechanism allows to balance short term/long term transitions. The background - the duality between distances between sequences and the evolution of the proximities between objects - is analysed.

Usage

```
seqemlt(seqdata, a = 1, b = 1, weighted = TRUE)
```

Arguments

seqdata	a state sequence object defined with the seqdef function.
a	optional argument for step weighing mechanism that controls the balancing between short term/long term transitions. The weighting function is $1/(a * s + b)$ where s is the transition step.
b	see argument a.
weighted	optional numerical vector containing weights, which may be used by some functions to compute weighted statistics (rates of transitions).

Details

The distance emlt between two sequences is the Euclidean distance between the transformed sequences coordinates. Using coord as the data input of any clustering algorithm using a Euclidean metric is equivalent of clustering with the emlt metric. A situation is defined as a state indexed with time, a sequence a timelines of states. The distance between situations is defined from the transitions between situations. The emlt distance between sequences takes into account the proximity between situations. Transitions are considered at any steps with a weighting balance between long/short terms. A situation may have no occurrence when the referring object is not present during all the duration. The distance between any situation and a situation with no occurrence is NA, and has no influence for the distance between sequences.

Value

An object of class emlt with the following componants

coord	transformed sequences. Euclidean metric emlt between sequences is equivalent to Euclidean distance between coord. coord is the input of any clustering algorithms using a Euclidean metric
states	list of states
situations	list of situations
sit.freq	frequence of situations
sit.transrate	rate of transitions from a situation to any situation of its own future : vector of transition towards future
sit.profil	profil of situations. The profil is a normalized vector issued from the rate of transition including a balance of short/long term with the weight of time 1/a*s+b, where s is the step of transition
sit.cor	Correlation between situations. Two situations are high correlated when their profiles are similar (ie their transitions towards future are similar).

Author(s)

Patrick Rousset, Senior researcher at Cereq, rousset@cereq.fr with the help of Matthias Studer

References

- Rousset Patrick, Giret Jean-françois,Classifying Qualitative Time Series with SOM: The Typology of Career Paths in France Lecture Notes in computer science, vol 4507, 2007, Springer Berlin / Heidelberg - - Rousset Patrick, Giret Jean-françois, Yvette Grelet (2012) Typologies De Parcours et Dynamique Longitudinale, Bulletin de méthodologie sociologique, issue 114, april 2012. - - Rousset Patrick, Giret Jean-françois (2008) A longitudinal Analysis of Labour Market Data with SOM, Encyclopedia of Artificial Intelligence, Edition Information Science Reference -

See Also

plot.emlt

Examples

```
data(mvad)
mvad.seq <- seqdef(mvad[1:100, 17:41])
alphabet(mvad.seq)
head(labels(mvad.seq))
## Computing distance
mvad.emlt <- seqemlt(mvad.seq)

## typology1 with kmeans in 3 clusters
km <- kmeans(mvad.emlt$coord, 3)

##Plotting typology1 by clusters
seqdplot(mvad.seq, group=km$cluster)

## typology2 : with ward criterion in 3 clusters for large data: a two step kmeans-cluster
km<-kmeans(mvad.emlt$coord,25)
hc<-hclust(dist(km$centers, method="euclidean"), method="ward")
zz<-cutree(hc, k=3)

##Plotting typology2 by clusters

seqdplot(mvad.seq, group=zz[km$cluster])


## Plotting the evolution of the correlation between states
plot(mvad.emlt, from="employment", to="joblessness",type="cor")
plot(mvad.emlt, from=c("employment","HE", "school", "FE"), to="joblessness", delay=0, leg=TRUE)
plot(mvad.emlt, from="joblessness", to="employment", delay=6)
plot(mvad.emlt, type="pca", cex=0.4, compx=1, compy=2)
```

`seqentrans` *Event sequence length and number of events*

Description

Adds the sequence length (number of transitions) and total number of events of event sequences to the data attribute of a `subseqelist` event sequence object.

Usage

```
seqentrans(fsubseq, avg.occ = FALSE)
```

Arguments

`fsubseq`	A `subseqelist` object as returned by `seqefsub`.
`avg.occ`	Should a column with average number of occurrences also be added.

Details

An event sequence object is an ordered list of transitions, with each transition a non-ordered list of events occurring at a same position.

Average occurrences by sequence may be useful when counts report number of occurrences rather than number of sequences containing the subsequence.

Value

The object `fsubseq` updated with the additional information.

Author(s)

Nicolas Müller and Gilbert Ritschard

Examples

```
data(actcal.tse)
actcal.seqe <- seqecreate(actcal.tse[1:500,])

##Searching for frequent subsequences appearing at least 30 times
fsubseq <- seqefsub(actcal.seqe, minSupport=10)
fsubseq <- seqentrans(fsubseq)
## dispaying only those with at least 3 transitions
fsubseq[fsubseq$data$ntrans>2]
## dispaying only those with at least 3 events
fsubseq[fsubseq$data$nevent>2]

## Average occurrences when counting distinct occurrences
ct <- seqeconstraint(countMethod="CDIST_O")
fsb <- seqefsub(actcal.seqe, minSupport=10, constraint=ct)
fsb <- seqentrans(fsb, avg.occ=TRUE)
fsb[1:10,]
```

seqeordplot *Plot the order of event sequences.*

Description

Illustrates the order of event sequences in a modified time-series plot. The x axis presents the position in the sequence the y axis the event.

Usage

```
seqeordplot(seqe,group=NULL,weighted=TRUE,weights=NULL,
            alphabet=NULL,
            type="distinctive",embedding="most-frequent",
            show=c(0,1),hide.col="grey75",
            cpal=NULL,alpha=1,
            lcourse="upwards",lorder="background",
            lweights=TRUE,lwd.min=0.5,lwd.max=4,lty=1,
            cex=1,border=grid.col,border.lwd=grid.lwd/2,
            grid.col="white",grid.fill="grey90",
            grid.scale=1/4,grid.shape="default",grid.lwd=0,
            orderalign=ifelse(split=="last","last","first"),
            split=NULL,layout=NULL,return.data=FALSE,
            main="",sub=NULL,mtext=TRUE,
            xlab="order position",ylab="",
            xlim=NULL,ylim=NULL,...)
```

Arguments

seqe	an event sequence object as defined by the seqecreate function.
group	A grouping vector.
weighted	Logial. Use of weights in the seqe object.
weights	A weight vector. Overwrites the weights in the seqe object.
alphabet	A vector with event names. Defines the arrangement in the y axis.
type	The trajectories type to draw. Either "distinctive" or "non-embeddable".
embedding	Option for type="non-embeddable", the method how to assign trajectories having multiple corresponding non-embeddable trajectories. Either "most-frequent" (default) or "uniformly".
show	Vector of two values between 0 and 1. Indicate the minimal and maximal relative frequency for a trajectory to be presented in the foreground of the plot.
hide.col	color of trimmed trajectories. If set to "white" trajectories are not shown at all.
cpal	A colour palette to be assigned to the sequences.
alpha	Degree of line and symbol transparency. Choose a number between 0 and 1.

lcourse	Handling for line connection of simultaneous observations. Either "upwards" or "downwards".
lorder	Either "background" (default) or "foreground". The first plots infrequent trajectories, the latter the frequent trajectories in the front.
lweights	logical. Should the line width be proportional to the represented trajectory? If 'FALSE', line width is set as "lwd.min".
lwd.min	The minimal line width to be drawn in the plot.
lwd.max	The maximal line width to be drawn in the plot.
lty	Line type of lines connecting succeeding observations.
cex	Expansion factor for the plotted squared symbols.
border	Color of symbol borders.
border.lwd	Line widths of symbol borders.
grid.col	Color of border of underlaid rectangles.
grid.fill	Color of underlaid rectangles.
grid.scale	Expansion degree of underlaid rectangles.
grid.shape	Either "default" or "proportional".
grid.lwd	Line width or border of underlaid rectangles.
orderalign	Alignment mode for data where the order positions are individual integer orders. align="first" aligns the trajectories left hand, align="last" right hand. Assigning an y category produces an alinement of the first occurrences of this category.
split	Logical value for plot panel arrangement modes. If 'TRUE' and 'x.orderalign="first"', the plot produces one plot panel per observed initial event and inscribes the trajectories which are initialised by that event. The case 'TRUE' and 'align="last"' proceeds analogously but considering final events.
layout	Integer vector of length 2. Determines the number of rows and columns of the plot panels arrangement.
return.data	Returns a summary of the plotted trajectories.
main	A main title for the plot, see also "title".
sub	Subtitles. Used in case of multiple plot panels.
mtext	Logical. Print panel information or not.
xlab	A label for the x axis, defaults to a description of "x".
ylab	A label for the y axis, defaults to a description of "y".
xlim	The x limits (x1, x2) of the plot.
ylim	The y limits (y1, y2) of the plot.
...	Arguments to be passed to methods, such as graphical parameters (see "par").

Examples

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

## loading the data and defining an event sequence dataset
## ========================================================

data(biofam)
lab <- c("Parent","Left","Married",
         "Left+Marr","Child","Left+Child",
         "Left+Marr+Child","Divorced")
biofam.seq <- seqdef(data=biofam[,10:25], alphabet=0:7, labels=lab)
## For this example, we consider only a sample of 200 cases
##set.seed(23653)
##sple <- sample(1:nrow(biofam.seq), size=200)
sple <- 500:700  ## need a sample with all elements of the alphabet
##seqstatl(biofam[sple,10:25])
biofam <- biofam[sple,]
biofam.seq <- biofam.seq[sple,]
bf.seqestate <- seqecreate(biofam.seq, tevent = "state")
head(bf.seqestate)

## plot the data
## =============

## distinctive event sequences

## Not run:
par(mar=c(4,8,2,2))
seqeordplot(seqe=bf.seqestate,alphabet=lab)

par(mar=c(4,8,2,2))
seqeordplot(seqe=bf.seqestate,alphabet=lab,
                      lwd.max=6,cex=0.9,show=c(0.05,1))

## non-embeddable sequences

par(mar=c(4,8,2,2))
seqeordplot(seqe=bf.seqestate,alphabet=lab,
                      lwd.max=6,cex=0.9,show=c(0.05,1),
                      type="non-embeddable")

## some additional options

par(mar=c(4,8,2,2)) # how the sequences end
seqeordplot(seqe=bf.seqestate,alphabet=lab,
                      type="non-embeddable",lwd.max=2,
                      orderalign="last",split="last")
```

```
par(mar=c(4,8,2,2)) # sequences involving Left+Marr+Child
seqeordplot(seqe=bf.seqestate,alphabet=lab,
                      type="non-embeddable",
                      orderalign="Left+Marr+Child")

par(mar=c(4,8,2,2)) # gender differences
seqeordplot(seqe=bf.seqestate,group=biofam$sex,show=c(0.05,1),
                      alphabet=lab)

## End(Not run)
```

`seqerulesdisc` *Extract association rules using discrete time regression models*

Description

Extract association rules from an object created by the `createdatadiscrete` function, using discrete time regression models to assess the significance of the extracted rules.

Usage

```
seqerulesdisc(fsubseq, datadiscr, tsef, pvalue=0.1, supvars=NULL,
   adjust=TRUE, topt=FALSE, link="cloglog", dep=NULL)
```

Arguments

`fsubseq`	an object created using the `seqefsub` function and that contains the list of subsequences to be tested for an association
`datadiscr`	the object created by the `createdatadiscrete` function and that contains the person-period data
`tsef`	the data frame containing the original time-to-event dataset (equivalent to the `data` argument from the `createdatadiscrete` function)
`pvalue`	the default threshold p-value to consider an association rule as significative, default is 0.1
`supvars`	a vector of variable names to be used as control variables in the regression models (experimental)
`adjust`	if set to TRUE, a Bonferroni adjustment is applied to the p-value threshold specified in the `pvalue` argument
`topt`	if set to TRUE, use an alternative algorithm to extract the rules (very experimental) ; default to FALSE
`link`	the link function to be used in the generalized linear regression model. To obtain hazard ratios, use the complementary log-log link function ("cloglog", as default). The other choice is to use a logit link function ("logit").
`dep`	if set to NULL, test all possible association rules. If an event is set, the function will only test association rules ending with this event

Details

This function uses a list of subsequences created by the `seqefsub` function from the TraMineR package and tests each possible association rules. It then shows the association rules whose significance, assessed using a discrete time regression model, is higher than the specified p-value threshold.

The algorithm is described in the Müller et al. (2010) article, even though this function uses a discrete time regression model instead of the Cox regression model described in the article. A more complete explanation of the method is available in Müller (2011).

Value

a list with one person-period data frame by event, where the dependent event is different each time. Please see the attached data file and code for an example.

Author(s)

Nicolas S. Müller

References

Müller, N.S., M. Studer, G. Ritschard et A. Gabadinho (2010), Extraction de règles d'association séquentielle à l'aide de modèles semi-paramétriques à risques proportionnels, *Revue des Nouvelles Technologies de l'Information*, **Vol. E-19**, EGC 2010, pp. 25-36.

Müller, N.S. (2011), Inégalités sociales et effets cumulés au cours de la vie : concepts et méthodes, *Thèse de doctorat, Faculté des sciences économiques et sociales, Université de Genève*, http://archive-ouverte.unige.ch/unige:17746.

See Also

`createdatadiscrete` to create the object needed as the `datadiscr` argument. `seqefsub` to create the object needed as the `fsubseq` argument.

Examples

```
##
```

`seqgen.missing` *Generate random missing states within a state sequence object*

Description

The function assigns missing values (`nr` attribute of the object, which is `"*"` by default) to randomly selected positions in randomly selected cases.

Usage

```
seqgen.missing(seqdata, p.cases = 0.1, p.left = 0.2, p.gaps = 0, p.right = 0.3,
          mt.left="nr", mt.gaps="nr", mt.right="nr")
```

Arguments

seqdata	A state sequence object
p.cases	Proportion of cases with missing values.
p.left	Proportion of left missing values.
p.gaps	Proportion of gap missing values.
p.right	Proportion of right missing values.
mt.left	Type of left missing. One of "nr" (non response state) or "void"
mt.gaps	Type of gap missing. One of "nr" (non response state) or "void"
mt.right	Type of right missing. One of "nr" (non response state) or "void"

Details

The aim of the function is essentially pedagogical. It may serve to illustrate how results of a sequential analysis may be affected by the presence of random missing states.

States in the sequences are randomly replaced with missing values. For each selected sequence, first, a random proportion between 0 and p.gaps of gaps are randomly inserted, then a random proportion between 0 and p.left of positions from the start of the sequence are set as missing, and finally a random proportion between 0 and p.right of positions from the end of the sequence are set as missing. Left missing values may possibly overlap gaps, and right missing values may overlap gaps and/or right missing values.

Value

The resulting state sequence object.

Warning

This function needs further testing.

Author(s)

Gilbert Ritschard

See Also

seqdef

Examples

```
## create the biofam.seq state sequence object from the biofam data.
data(biofam)
biofam.seq <- seqdef(biofam[1:100,10:25])

## Generate missing states within 60% of the sequences.
biofamm.seq <- seqgen.missing(biofam.seq, p.cases=.6,
      p.left=.4, p.gaps=.2, p.right=.5)
```

```
## compare the rendering of the sequences before and after
##   introducing missing states.
par(mfrow=c(2,2))
seqIplot(biofam.seq, sortv="from.end", withlegend=FALSE)
seqIplot(biofamm.seq, sortv="from.end", withlegend=FALSE)
seqdplot(biofam.seq, with.missing=TRUE, border=NA, withlegend=FALSE)
seqdplot(biofamm.seq, with.missing=TRUE, border=NA, withlegend=FALSE)
dev.off()
```

`seqgranularity` *Changing sequence time granularity by aggregating positions*

Description

Changes time granularity of a state sequence object by aggregating successive positions into groups of a user-defined time length.

Usage

```
seqgranularity(seqdata, tspan = 3, method = "last")
```

Arguments

`seqdata`	A state sequence object.
`tspan`	Integer. Number of successive positions grouped together.
`method`	Character string. Aggregating method. One of `"first"` or `"last"` (default).

Details

Successive position are aggregated by group of `tspan`, starting from the first position and, with method `"last"`, the last state in the group is assigned to the group. In case of an incomplete last group, the last state is assigned to it.

Value

The resulting state sequence object.

Warning

This function needs further testing.

Author(s)

Matthias Studer and Gilbert Ritschard

See Also

`seqdef`

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

data(mvad)
mvad <- mvad[1:100,]
mvad.seq <- seqdef(mvad[,17:86], xtstep=12)
mvadg.seq <- seqgranularity(mvad.seq, tspan=6)
par(mfrow=c(2,1))
seqdplot(mvad.seq, withlegend=FALSE, border=NA)
seqdplot(mvadg.seq, withlegend=FALSE)
```

---

seqplot.tentrop *Plotting superposed transversal-entropy curves*

Description

Functions to plot, in a same frame, transversal-entropy curves by group or multiple curves.

Usage

```
seqplot.tentrop(seqdata, group, title=NULL,
    col=NULL, lty=NULL, lwd=3.5, ylim=NULL, xtlab=NULL,
    xtstep=NULL, withlegend=TRUE, glabels=NULL,
    legendpos="topright", horiz=FALSE, cex.legend=1, ...)

seqplot.tentrop.m(seqdata.list, title=NULL,
    col=NULL, lty=NULL, lwd=3.5, ylim=NULL, xtlab=NULL,
    xtstep=NULL, withlegend=TRUE, glabels=NULL,
    legendpos="topright", horiz=FALSE, cex.legend=1, ...)
```

Arguments

seqdata	a state sequence object (see seqdef).
seqdata.list	a list of state sequence objects.
group	a factor giving the group membership of each sequence in seqdata.
title	a character string giving the title of the graphic; if NULL, a default title is printed.
col	a vector of colors for the different curves.
lty	a vector of line types for the different curves. See lines.
lwd	numeric or vector of numerics: width of curve lines. See lines.
ylim	pair of numerics defining the range for the y-axis. If left NULL, the limits are defined from the data.
xtlab	vector of strings defining the x-axis tick labels.
xtstep	integer: step between tick marks on the x-axis.

glabels	a vector of strings with the curve labels. If NULL curves are labeled with the levels of the group variable or, for seqplot.tentrop.m, as seq1, seq2, ...
withlegend	logical: Should the legend be plotted. Default is TRUE.
legendpos	legend position: default is "topright". Seelegend .
horiz	logical: Should the legend be displayed horizontally. Set as FALSE by default, i.e., legend is displayed vertically.
cex.legend	Scale factor for the legend.
...	additional plot parameters (see par).

Details

Use seqplot.tentrop to plot curves of transversal entropies by groups of a same set of sequences, e.g. professional careers by sex.

Use seqplot.tentrop.m to plot multiple curves of transversal entropies corresponding to different sets of sequences such as sequences describing cohabitational and sequences describing occupational trajectories.

See Also

seqHtplot for an alternative way of plotting the transversal entropies and seqstatd to get the values of the entropies.

Examples

```
## Using the biofam data which has sequences from
## ages 15 to 30 years in columns 10 to 25
data(biofam)
biofam <- biofam[1:200,]
biofam.seq <- seqdef(biofam[,10:25], xtlab=as.character(15:30), xtstep=3)

## Plotting transversal entropies by sex
seqplot.tentrop(biofam.seq, group=biofam$sex, legendpos="bottomright")

## Plotting transversal entropies for women
## by father's social status
group <- biofam$cspfaj[biofam$sex=="woman"]
seqplot.tentrop(biofam.seq[biofam$sex=="woman",], group=group,
    title="Women, by father's social status", legendpos="bottomright")
```

---

seqrep.grp *Finding representative sets by group and their quality statistics.*

Description

This function determines representative sequences by group and returns the representatives by group and/or the quality statistics of the representative sets. The function is a wrapper for the TraMineR seqrep function.

Usage

```
seqrep.grp(seqdata, group = NULL, mdis = NULL, ret="stat", ...)
```

Arguments

seqdata	state sequence object as defined by seqdef.
group	group variable. If NULL a single group is assumed.
mdis	dissimilarity matrix. If NULL the "LCS" dissimilarity matrix is computed.
ret	What should be returned? One of "stat" (default), "rep" or "both".
...	additional arguments passed to seqrep .

Details

The function is a wrapper for running seqrep on the different groups defined by the group variable.

Value

If ret="stat", a list with the quality statistics for the set of representatives of each group.

If ret="rep", a list with the set of representatives of each group.

If ret="both", a list with the two previous outcomes.

Note

This function is a pre-release and further testing is still needed, please report any problems.

Author(s)

Gilbert Ritschard

See Also

seqrep

Examples

```
data(biofam)
biofam <- biofam[1:100,]
biofam.lab <- c("Parent", "Left", "Married", "Left+Marr",
"Child", "Left+Child", "Left+Marr+Child", "Divorced")
biofam.short <- c("P","L","M","LM","C","LC","LMC","D")
biofam.seq <- seqdef(biofam[,10:25], alphabet=0:7, states=biofam.short, labels=biofam.lab)
dist <- seqdist(biofam.seq, method="HAM", with.missing=TRUE)

seqrep.grp(biofam.seq, group=biofam$plingu02, mdis=dist, trep=.2, tsim=.1)
seqrep.grp(biofam.seq, group=biofam$plingu02, mdis=dist, ret="rep", trep=.2, tsim=.1)
```

seqstart *Aligning sequence data on a new start time.*

Description

Changing the position alignment of a set of sequences.

Usage

```
seqstart(seqdata, data.start, new.start, tmin = NULL, tmax = NULL, missing = NA)
```

Arguments

seqdata	a data frame or matrix containing sequence data.
data.start	Integer. The actual starting date of the sequences. In case of sequence-dependent start dates, should be a vector of length equal to the number of rows of seqdata.
new.start	Integer. The new starting date. In case of sequence-dependent start dates, should be a vector of length equal to the number of rows of seqdata.
tmin	Integer. Start position on new position axis. If NULL, it is guessed from the data.
tmax	Integer. End position on new position axis. If NULL, it is guessed from the data.
missing	Character. Code used to fill missing data in the new time axis.

Value

A matrix.

Note

Warning: This function needs further testing.

Author(s)

Matthias Studer

Examples

```
#An example data set
paneldata <- matrix(c("A" ,"A" , "B" , "B" , "B",
"A" , "A" , "B" , "B" , "B",
"A" , "A",  "B" , "B" , "B" ,
"A"  ,"A" , "A" , "B"  ,"B" ,
"A"  ,"A" , "A" , "A" , "B"), byrow=TRUE, ncol=5)
colnames(paneldata) <- 2000:2004

print(paneldata)
```

```
## Assuming data are aligned on calendar years, starting in 2000
## Change from calendar date to age alignment
startyear <- 2000
birthyear <- 1995:1999
agedata <- seqstart(paneldata, data.start=startyear, new.start=birthyear)
colnames(agedata) <- 1:ncol(agedata)
print(agedata)

## Retaining only ages between 3 and 7 (4th and 8th year after birthyear).
seqstart(paneldata, data.start=startyear, new.start=birthyear, tmin=4, tmax=8, missing="*")

## Changing back from age to calendar time alignment
ageatstart <- startyear - birthyear
seqstart(agedata, data.start=1,  new.start=ageatstart)
## Same but dropping right columns filled with NA's
seqstart(agedata, data.start=1,  new.start=ageatstart, tmax=5)
```

`sortv` *Sort sequences by states at the successive positions*

Description

Returns a sorting vector to sort state sequences in a TraMineR sequence object (`seqdef`) by the states at the successive positions.

Usage

```
sorti(seqdata, start = "end", sort.index=TRUE)

sortv(seqdata, start = "end")
```

Arguments

`seqdata`	A state sequence object as returned by `seqdef`.
`start`	Where to start the sort. One of `"beg"` (beginning) or `"end"`.
`sort.index`	Should the function return sort indexes? If `FALSE`, sort values are returned.

Details

With `start = "end"` (default), the primary sort key is the final state, then the previous one and so on. With `start = "beg"`, the primary sort key is the state at the first position, then at the next one and so on.

With `sort.index = FALSE`, the function returns a vector of values whose order will determine the wanted order. This should be used as `sortv` argument of the `seqiplot` function. With `sort.index = TRUE`, the function returns a vector of indexes to be used for indexing.

The `sortv` form is an alias for `sorti(..., sort.index = FALSE)`.

Value

If `sort.index = FALSE`, the vector of sorting values.
Otherwise the vector of sorting indexes.

Author(s)

Gilbert Ritschard

See Also

Details about `type = "i"` or `type = "I"` in `seqplot` .

Examples

```
data(actcal)
actcal.seq <- seqdef(actcal[1:100,13:24])
par(mfrow=c(1,2))
seqIplot(actcal.seq, sortv=sortv(actcal.seq), withlegend = FALSE)
seqIplot(actcal.seq, sortv=sortv(actcal.seq, start="beg"), withlegend = FALSE)
actcal.seq[sorti(actcal.seq)[90:100],]

data(mvad)
mvad.seq <- seqdef(mvad[1:100,17:86])
par(mfrow=c(1,2))
seqIplot(mvad.seq, sortv=sortv(mvad.seq, start="end"), withlegend = FALSE)
seqIplot(mvad.seq, sortv=sortv(mvad.seq, start="beg"), withlegend = FALSE)
print( mvad.seq[sorti(mvad.seq,start="beg")[90:100],], format="SPS")
```

STS_to_SPELL *Data conversion from STS to SPELL format.*

Description

Convert data from STS to vertical SPELL format.

Usage

```
STS_to_SPELL(seqdata, id=NULL, pdata=NULL, birthdate=NULL, with.missing=TRUE)
```

Arguments

`seqdata`	a state sequence object of type `"stslist"` (see `seqdef`).
`id`	Either a vector of sequence id's or name of the id column in `pdata`. If `NULL`, `rownames(seqdata)` are used.
`pdata`	Data frame with id and birth dates.
`birthdate`	Either a vector of sequence birth dates or name of the birth date column in `pdata`.
`with.missing`	Logical. Should explicit spells of missing states be included?

Details

SPELL format is a vertical format with one row per spell, each spell being specified with four variables: an id, a begin date, a end date, and the state.

Value

A `data.frame` with the sequences in vertical SPELL format.

Note

This function is a pre-release and further testing is still needed, please report any problems.

Author(s)

Matthias Studer

See Also

See Also `seqformat`.

Examples

```
data(biofam)
biofam <- biofam[1:20,]

## Create the sequence object
bfstates <- c("Parent", "Left", "Married", "Left/Married", "Child",
            "Left/Child", "Left/Married/Child", "Divorced")
bf.shortlab <- c("P","L","M","LM","C","LC", "LMC", "D")
bf.seq <- seqdef(biofam[,10:25], alphabet=0:7, states=bf.shortlab, labels=bfstates)

spell <- STS_to_SPELL(bf.seq, birthdate=biofam$birthyr)
head(spell)
```

---

toPersonPeriod *Converting into person-period format.*

Description

Converts the STS sequences of a state sequence object into person-period format.

Usage

```
toPersonPeriod(seqdata)
```

Arguments

`seqdata`	A state sequence object as returned by `seqdef`.

Value

A data frame with three columns: `id`, `state` and `timestamp`.

Author(s)

Matthias Studer

See Also

`seqformat` .

Examples

```
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)

mvad2 <- toPersonPeriod(mvad.seq[1:20,])
```

TSE_to_STS *Converting TSE data into STS (state sequences) format.*

Description

Conversion from TSE (time stamped event sequences) vertical format into STS (state sequences) data format.

Usage

```
TSE_to_STS(seqdata, id = 1, timestamp = 2, event = 3, stm = NULL, tmin = 1,
    tmax = NULL, firstState = "None")
```

Arguments

`seqdata`	a data frame or matrix with event sequence data in TSE format.
`id`	Name or index of the column containing the id's of the sequences.
`timestamp`	Name or index of the column containing the timestamps of the events.
`event`	Name or index of the column containing the events.
`stm`	An event to state transition matrix (See `seqe2stm`).
`tmin`	Integer. Starting time of the state sequence.
`tmax`	Integer. Ending time of the state sequence.
`firstState`	Character. The name of the state before any events has occurred.

Details

Convert TSE (time stamped event sequences) data into STS (state sequences) format. By default, the states are defined has the combination of events that already occurred. Different schemes may be specified using function `seqe2stm` and the `stm` argument.

Value

A `data.frame` with the sequences in STS format.

Note

This function is a pre-release and further testing is still needed, please report any problems.

Author(s)

Matthias Studer

References

Ritschard, G., Gabadinho, A., Studer, M. & Müller, N.S. (2009), "Converting between various sequence representations", In Ras, Z. & Dardzinska, A. (eds) *Advances in Data Management*. Series: *Studies in Computational Intelligence*. Volume 223, pp. 155-175. Berlin: Springer.

See Also

See Also `seqe2stm`, `seqformat`.

Examples

```
data(actcal.tse)
events <- c("PartTime", "NoActivity", "FullTime", "LowPartTime")
## Dropping all previous events.
stm <- seqe2stm(events, dropList=list("PartTime"=events[-1],
   NoActivity=events[-2], FullTime=events[-3], LowPartTime=events[-4]))
mysts <- TSE_to_STS(actcal.tse[1:100,], id=1, timestamp=2, event=3,
   stm=stm, tmin=1, tmax=12, firstState="None")
```

Index