Package 'PST'

July 21, 2025

Version 0.94.1

Date 2017-02-02

Title Probabilistic Suffix Trees and Variable Length Markov Chains

Author Alexis Gabadinho [aut, cre, cph]

Maintainer Alexis Gabadinho <alexis.gabadinho@wanadoo.fr>

Depends R (>= 2.10), TraMineR, RColorBrewer

Imports methods, stats4

Description Provides a framework for analysing state sequences with probabilistic suffix trees (PST), the construction that stores variable length Markov chains (VLMC). Besides functions for learning and optimizing VLMC models, the PST library includes many additional tools to analyse sequence data with these models: visualization tools, functions for sequence prediction and artificial sequences generation, as well as for context and pattern mining. The package is specifically adapted to the field of social sciences by allowing to learn VLMC models from sets of individual sequences possibly containing missing values, and by accounting for case weights. The library also allows to compute probabilistic divergence between two models, and to fit segmented VLMC, where sub-models fitted to distinct strata of the learning sample are stored in a single PST. This software results from research work executed within the framework of the Swiss National Centre of Competence in Research LIVES, which is financed by the Swiss National Science Foundation. The authors are grateful to the Swiss National Science Foundation for its financial support.

License GPL (>= 2)

URL https://r-forge.r-project.org/projects/pst

Repository CRAN

Repository/R-Forge/Project pst

Repository/R-Forge/Revision 297

Repository/R-Forge/DateTimeStamp 2017-02-02 16:41:19

Date/Publication 2023-12-14 15:52:30 UTC

NeedsCompilation no

Contents

`cmine` *Mining contexts*

Description

Extracting contexts in a PST satisfying user defined criterion

Usage

```
## S4 method for signature 'PSTf'
cmine(object, l, pmin, pmax, state, as.tree=FALSE, delete=TRUE)
```

Arguments

object	A probabilistic suffix tree, i.e., an object of class "PSTf" as returned by the pstree, prune or tune function.
l	length of the context to search for.
pmin	numeric. Minimal probability for selecting the (sub)sequence.
pmax	numeric. Maximal probability for selecting the (sub)sequence.
state	character. One or several states of the alphabet for which the (cumulated) probability is greater than pmin or less than pmax.
as.tree	logical. If TRUE the cmine method returns a subtree of the PST given as input with selected contexts (including their parent nodes, even if these don't statistify the defined criterion). If FALSE the output is the list of selected contexts. See value.
delete	Logical. If as.tree=TRUE and delete=FALSE, the pruned nodes are not removed from the tree but tagged as pruned=FALSE, so that when plotting the pruned tree these nodes wil appear surrounded with red (can be set to another color) lines.

Value

If as.tree=TRUE a PST, that is an object of class PSTf which can be printed and plotted; if as.tree=FALSE a list of contexts with their associated next symbol probability distribution, that is an object of class cprobd.list for which a plot method is available. Subscripts can be used to select subsets of the contexts, see examples.

details

The cmine function searches in the tree for nodes fulfilling certain characteristics, for example contexts that are highly likely to be followed by a given state (see example 1). One can also mine for contexts corresponding to a minimum or maximum probability for several states together (see example 2). For more details, see *Gabadinho 2016*.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

Examples

```
## Loading the SRH.seq sequence object
data(SRH)

## Learning the model
SRH.pst <- pstree(SRH.seq, nmin=30, ymin=0.001)
```

```
## Example 1: searching for all contexts yielding a probability of the
## state G1 (very good health) of at least pmin=0.5
cm1 <- cmine(SRH.pst, pmin=0.5, state="G1")
cm1[1:10]

## Example 2: contexts associated with a high probability of
## medium or lower self rated health
cm2 <- cmine(SRH.pst, pmin=0.5, state=c("B1", "B2", "M"))
plot(cm2, tlim=0, main="(a) p(B1,B2,M)>0.5")
```

`cplot` *Plot single nodes of a probabilistic suffix tree*

Description

Plot the next symbol probability distribution associated with a particular node in a PST

Usage

```
## S4 method for signature 'PSTf'
cplot(object, context, state, main=NULL, all=FALSE, x.by=1, y.by=0.2, by.state=FALSE, ...)
```

Arguments

`object`	A probabilistic suffix tree, i.e., an object of class `"PSTf"` as returned by the `pstree`, `prune` or `tune` function.
`context`	character. Label of the node to plot, provided as a string where states are separated by '-', see examples.
`state`	logical. Under development.
`main`	character. Main title for the plot. By default, the title is the node label.
`all`	logical.
`x.by`	numeric. Interval for the ticks on the x axis (segments).
`y.by`	numeric. Interval for the ticks on the y axis (probability).
`by.state`	logical. If `TRUE`, the representation of the probability distribution is done separately for each state of the alphabet.
`...`	arguments to be passed to the plot function or other graphical parameters.

Details

The `cplot()` function displays a single node labelled with `context` of the tree where one or mode barplots (if `object` is a segmented PST) represent the probability distribution(s) stored in the node. For more details, see *Gabadinho 2016*.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

See Also

ppplot

Examples

```
data(s1)
s1 <- seqdef(s1)
S1 <- pstree(s1, L=3)

cplot(S1, "a-b")
```

---

cprob *Empirical conditional probability distributions of order* L

Description

Compute the empirical conditional probability distributions of order L from a set of sequences

Usage

```
## S4 method for signature 'stslist'
cprob(object, L, cdata=NULL, context, stationary=TRUE, nmin=1, prob=TRUE,
weighted=TRUE, with.missing=FALSE, to.list=FALSE)
```

Arguments

object	a sequence object, that is an object of class stslist as created by TraMineR seqdef function.
L	integer. Context length.
cdata	under development
context	character. An optional subsequence (a character string where symbols are separated by '-') for which the conditional probability distribution is to be computed.
stationary	logical. If FALSE probability distributions are computed for each sequence position L+1 ... l where l is the maximum sequence length. If TRUE the probability distributions are stationary that is time homogenous.
nmin	integer. Minimal frequency of a context. See details.
prob	logical. If TRUE the probability distributions are returned. If FALSE the function returns the empirical counts on which the probability distributions are computed.
weighted	logical. If TRUE case weights attached to the sequence object are used in the computation of the probabilities.

with.missing	logical. If FALSE only contexts contining no missing status are considered.
to.list	logical. If TRUE and stationary=TRUE, a list instead of a matrix is returned. See value.

Details

The empirical conditional probability $\hat{P}(\sigma|c)$ of observing a symbol $\sigma \in A$ after the subsequence $c = c_1, \ldots, c_k$ of length $k = L$ is computed as

$$\hat{P}(\sigma|c) = \frac{N(c\sigma)}{\sum_{\alpha \in A} N(c\alpha)}$$

where

$$N(c) = \sum_{i=1}^{\ell} 1\left[x_i, \ldots, x_{i+|c|-1} = c\right],\ x = x_1, \ldots, x_\ell,\ c = c_1, \ldots, c_k$$

is the number of occurrences of the subsequence c in the sequence x and $c\sigma$ is the concatenation of the subsequence c and the symbol σ.

Considering a - possibly weighted - sample of m sequences having weights $w^j,\ j = 1 \ldots m$, the function $N(c)$ is replaced by

$$N(c) = \sum_{j=1}^{m} w^j \sum_{i=1}^{\ell} 1\left[x_i^j, \ldots, x_{i+|c|-1}^j = c\right],\ c = c_1, \ldots, c_k$$

where $x^j = x_1^j, \ldots, x_\ell^j$ is the jth sequence in the sample. For more details, see *Gabadinho 2016*.

Value

If stationary=TRUE a matrix with one row for each subsequence of length L and minimal frequency $nmin$ appearing in object. If stationary=FALSE a list where each element corresponds to one subsequence and contains a matrix whith the probability distribution at each position p where a state is preceded by the subsequence.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

Examples

```
## Example with the single sequence s1
data(s1)
s1 <- seqdef(s1)
cprob(s1, L=0, prob=FALSE)
cprob(s1, L=1, prob=TRUE)
```

```
## Preparing a sequence object with the SRH data set
data(SRH)
state.list <- levels(SRH$p99c01)
## sequential color palette
mycol5 <- rev(brewer.pal(5, "RdYlGn"))
SRH.seq <- seqdef(SRH, 5:15, alphabet=state.list, states=c("G1", "G2", "M", "B2", "B1"),
labels=state.list, weights=SRH$wp09lp1s, right=NA, cpal=mycol5)
names(SRH.seq) <- 1999:2009

## Example 1: 0th order: weighted and unweigthed counts
cprob(SRH.seq, L=0, prob=FALSE, weighted=FALSE)
cprob(SRH.seq, L=0, prob=FALSE, weighted=TRUE)

## Example 2: 2th order: weighted and unweigthed probability distrib.
cprob(SRH.seq, L=2, prob=TRUE, weighted=FALSE)
cprob(SRH.seq, L=2, prob=TRUE, weighted=TRUE)
```

`generate` *Generate sequences using a probabilistic suffix tree*

Description

Generate sequences using a probabilistic suffix tree

Usage

```
## S4 method for signature 'PSTf'
generate(object, l, n, s1, p1, method, L, cnames)
```

Arguments

`object`	a probabilistic suffix tree, i.e., an object of class "`PSTf`" as returned by the `pstree`, `prune` or `tune` function.
`l`	integer. Length of the sequence(s) to generate.
`n`	integer. Number of the sequence(s) to generate.
`s1`	character. The first state in the sequences. The length of the vector should equal `n`. If specified, the first state in the sequence(s) is not randomly generated but taken from `s1`.
`p1`	numeric. An optional probability vector for generating the first position state in the sequence(s). If specified, the first state in the sequence(s) is randomly generated using the probability distribution in `p1` instead of the probability distribution taken fron the root node of `object`.
`method`	character. If `method=pmax`, at each position the state having the highest probability is chosen. If `method=prob`, at each position the state is generated using the corresponding probability distribution taken from `object`.
`L`	integer: Maximal depth used to extract the probability distributions from the PST object.

`cnames` character: Optional column (position) names for the returned state sequence object. By default, the names of the sequence object to which the model was fitted are used (slot "data" of the PST).

Details

As a probabilistic suffix tree (PST) represents a generating model, it can be used to generate artificial sequence data sets. Sequences are built by generating the states at each successive position. The process is similar to sequence prediction (see `predict`), except that the retrieved conditional probability distributions provided by the PST are used to generate a symbol instead of computing the probability of an existing state. For more details, see *Gabadinho 2016*.

Value

A state sequence object (an object of class `stslist`) containing `n` sequences. This object can be passed as argument to all the functions for visualization and analysis provided by the `TraMineR` package.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

Examples

```
data(s1)
s1.seq <- seqdef(s1)
S1 <- pstree(s1.seq, L=3)

## Generating 10 sequences
generate(S1, n=10, l=10, method="prob")

## First state is generated with p(a)=0.9 and p(b)=0.1
generate(S1, n=10, l=10, method="prob", p1=c(0.9, 0.1))
```

---

impute *Impute missing values using a probabilistic suffix tree*

Description

Missing states in a set of sequences are imputed by using the probability distributions stored in a probabilistic suffix tree.

Usage

```
## S4 method for signature 'PSTf,stslist'
impute(object, data, method="pmax")
```

Arguments

object	a probabilistic suffix tree, i.e., an object of class "PSTf" as returned by the pstree, prune or tune function.
data	a sequence object, i.e., an object of class 'stslist' as created by TraMineR seqdef function, containing the sequences to impute. See details.
method	character. If method='pmax' the state having the highest probability according to the probability distribution associated with the context preceding the missing status is imputed. If method='prob' the imputation is done randomly by using this probability distribution.

Details

A probabilistic suffix tree (PST) can be used to impute missing states in sequences built on the same alphabet. When a missing state occurs in a sequence the procedure searches in the PST for the context preceding the missing state and impute the state according to the conditional distribution associated with the context. The imputation can be done either randomly (method="prob") or with the state having the highest probability. However, more sophisticated modelling taking account of the non response mechanism could be required for imputing missing states. For more details, see *Gabadinho 2016*.

Value

A sequence object (of class stslist) containing original sequences in data with missing states imputed.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, 2016, **72**(3), 1-39.

Examples

```
## Loading the SRH.seq sequence object
data(SRH)

## working with a sub-sample of 500 sequences
## to reduce computing time
subs <- sample(nrow(SRH.seq), size=500)
SRH.sub.seq <- SRH.seq[subs,]
```

```
## Learning the model (missing state is not included)
SRH.pst.L10 <- pstree(SRH.sub.seq, nmin=2, ymin=0.001)

## Pruning
C99 <- qchisq(0.99,5-1)/2
SRH.pst.L10.C99 <- prune(SRH.pst.L10, gain="G2", C=C99)

## Imputing missing values in the SRH sequences
SRH.sub.iseq <- impute(SRH.pst.L10, SRH.sub.seq, method="prob")

## locating sequences having missing values
## in sequence object missing states are identified by '*'
have.miss <- which(rowSums(SRH.sub.seq=='*')>0)

## plotting non imputed vs imputed sequence
## (first 10 sequences in the set)
par(mfrow=c(1,2))
seqiplot(SRH.sub.seq[have.miss,], withlegend=FALSE)
seqiplot(SRH.sub.iseq[have.miss,], withlegend=FALSE)
```

`logLik` *Log-Likelihood of a variable length Markov chain model*

Description

Retrieve the log-likelihood of a fitted VLMC. This is the `logLik` method for objects of class `PSTf` returned by the `pstree` and `prune` functions.

Usage

```
## S4 method for signature 'PSTf'
logLik(object)
```

Arguments

`object`	a probabilistic suffix tree, i.e., an object of class `"PSTf"` as returned by the `pstree`, `prune` or `tune` function.

Details

The likelihood of a learning sample containing n sequences, given a model S fitted to it, is

$$L(S) = \prod_{i=1}^{n} P^S(x^i)$$

where $P^S(x^i)$ is the probability of the ith observed sequence predicted by S. Note that the log-likelihood of a VLMC model is not used in the estimation of the model's parameters (see `pstree`). It is obtained once the model is estimated by calling the `predict` function. The value is stored in the `logLik` slot of the probabilistic suffix tree representing the model (a `PSTf` object returned by the `pstree` or `prune` function). The `AIC` and `BIC` values can also be obtained with the corresponding generic functions, which call `logLik` and use its result. For more details, see *Gabadinho 2016*.

Value

An object of class `logLik`, a negative numeric value with the `df` (degrees of freedom) attribute containing the number of free parameters of the model.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

See Also

AIC, BIC

Examples

```
## activity calendar for year 2000
## from the Swiss Household Panel
## see ?actcal
data(actcal)

## selecting individuals aged 20 to 59
actcal <- actcal[actcal$age00>=20 & actcal$age00 <60,]

## defining a sequence object
actcal.lab <- c("> 37 hours", "19-36 hours", "1-18 hours", "no work")
actcal.seq <- seqdef(actcal,13:24,labels=actcal.lab)

## building a PST
actcal.pst <- pstree(actcal.seq, nmin=2, ymin=0.001)
logLik(actcal.pst)

## Cut-offs for 5% and 1% (see ?prune)
C99 <- qchisq(0.99,4-1)/2

## pruning
actcal.pst.C99 <- prune(actcal.pst, gain="G2", C=C99)

## Comparing AIC
AIC(actcal.pst, actcal.pst.C99)
```

nobs *Extract the number of observations to which a VLMC model is fitted*

Description

The number of observations to which a VLMC model is fitted is notably used for computing the Bayesian information criterion BIC or the Akaike information criterion with correction for finite sample sizes AICc.

Usage

```
## S4 method for signature 'PSTf'
nobs(object)
```

Arguments

object	A PST, that is an object of class PSTf as returned by the pstree or prune method.

Details

This is the method for the generic nobs function provided by the stats4 package. The number of observations to which a VLMC model is fitted is the total number of symbols in the learning sample. If the learning sample contains missing values and the model is learned without including missing values (see pstree), the total number of symbols is the number of non-missing states in the sequence(s). This information is used to compute the Bayesian information criterion of a fitted VLMC model. The BIC generic function calls the logLik and nobs methods for class PSTf. For more details, see *Gabadinho 2016*.

Value

An integer containing the number of symbols in the learning sample.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

See Also

BIC

Examples

```
data(s1)
s1.seq <- seqdef(s1)
S1 <- pstree(s1.seq, L=3)
nobs(S1)

## Self rated health sequences
## Loading the 'SRH' data frame and 'SRH.seq' sequence object
data(SRH)

## model without considering missing states
## model with max. order 2 to reduce computing time
## nobs is the same whatever L and nmin
m1 <- pstree(SRH.seq, L=2, nmin=30, ymin=0.001)
nobs(m1)

## considering missing states, hence nobs is higher
m2 <- pstree(SRH.seq, L=2, nmin=30, ymin=0.001, with.missing=TRUE)
nobs(m2)
```

---

nodenames *Retrieve the node labels of a PST*

Description

Retrieve the node labels of a PST

Usage

```
## S4 method for signature 'PSTf'
nodenames(object, L)
```

Arguments

object	A PST, that is an object of class PSTf as returned by the pstree or prune method.
L	integer. Depth of the tree for which the node names are retrieved. If missing the names of all the nodes in the tree are returned.

Value

A vector containing the node labels (i.e. contexts).

Author(s)

Alexis Gabadinho

Examples

```
data(s1)
s1 <- seqdef(s1)
S1 <- pstree(s1, L=3)

nodenames(S1, L=3)
nodenames(S1)
```

---

pdist *Compute probabilistic divergence between two PST*

Description

Compute probabilistic divergence between two PST

Usage

```
## S4 method for signature 'PSTf,PSTf'
pdist(x,y, method="cp", l, ns=5000, symetric=FALSE, output="all")
```

Arguments

x	a probabilistic suffix tree, i.e., an object of class "PSTf" as returned by the pstree, prune or tune function.
y	a probabilistic suffix tree, i.e., an object of class "PSTf" as returned by the pstree, prune or tune function.
method	character. Method for computing distances. So far only one method is available.
l	integer. Length of the sequence(s) to generate.
ns	integer. Number sequences to generate.
symetric	logical. If TRUE, the symetric version of the measure is returned, see details.
output	character. See value.

Details

The function computes a probabilistic divergence measure between PST S_A and S_B based on the measure originally proposed in *Juang-1985* and *Rabiner-1989* for the comparison of two (hidden) Markov models S_A and S_B

$$d(S_A, S_B) = \frac{1}{\ell}[\log P^{S_A}(x) - \log P^{S_B}(x)] = \frac{1}{\ell} \log \frac{P^{S_A}(x)}{P^{S_B}(x)}$$

where $x = x_1, \ldots, x_\ell$ is a sequence generated by model S_A, $P^{S_A}(x)$ is the probability of x given model S_A and $P^{S_B}(x)$ is the probability of x given model S_B. The ratio between the two sequence likelihoods measures how many times the sequence x is more likely to have been generated by S_A than by S_2.

As the number n of generated sequences on which the measure is computed (or the length of a single sequence) approaches infinity, the expected value of $d(S_A, S_B)$ converges to $d_{KL}(S_A, S_B)$ *Falkhausen-1995*, *He-2000*, the Kullback-Leibler (KL) divergence (also called information gain) used in information theory to measure the difference between two probability distributions.

The `pdist` function uses the following procedure to compute the divergence between two PST:

- generate a ransom sample of n sequences (of length ℓ) with model S_A using the `generate` method
- predict the sequences with S_A and with S_B
- compute
$$d_i(S_A, S_B) = \frac{1}{\ell}[\log P^{S_A}(x_i) - \log P^{S_B}(x_i))], \ i = 1, \ldots, n$$
- the expected value
$$E(d(S_A, S_B))$$
is the divergence between models S_A and S_B and is estimated as
$$\hat{E}(d(S_A, S_B)) = \frac{1}{n}\sum_{i=1}^{n} d_i(S_A, S_B)$$

For more details, see *Gabadinho 2016*.

Value

If `ouput="all"`, a vector containing the divergence value for each generated sequence, if `output="mean"`, the mean, i.e. expected value which is the divergence between models.

Author(s)

Alexis gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

Juang, B. H. and Rabiner, L. R. (1985). A probabilistic distance measure for hidden Markov models. *ATT Technical Journal*, **64**(2), pp. 391-408.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2), pp. 257-286.

Examples

```
## activity calendar for year 2000
## from the Swiss Household Panel
## see ?actcal
data(actcal)

## selecting individuals aged 20 to 59
actcal <- actcal[actcal$age00>=20 & actcal$age00 <60,]
```

```
## defining a sequence object
actcal.lab <- c("> 37 hours", "19-36 hours", "1-18 hours", "no work")
actcal.seq <- seqdef(actcal,13:24,labels=actcal.lab)

## building a PST segmented by age group
gage10 <- cut(actcal$age00, c(20,30,40,50,60), right=FALSE,
labels=c("20-29","30-39", "40-49", "50-59"))

actcal.pstg <- pstree(actcal.seq, nmin=2, ymin=0.001, group=gage10)

## pruning
C99 <- qchisq(0.99,4-1)/2
actcal.pstg.opt <- prune(actcal.pstg, gain="G2", C=C99)

## extracting PST for age group 20-39 and 30-39
g1.pst <- subtree(actcal.pstg.opt, group=1)
g2.pst <- subtree(actcal.pstg.opt, group=2)

## generating 5000 sequences with g1.pst
## and computing 5000 distances
dist.g1_g2 <- pdist(g1.pst, g2.pst, l=11)
hist(dist.g1_g2)

## the probabilistic distance is the mean
## of the 5000 distances
mean(dist.g1_g2)
```

---

plot-PSTr *Plot a PST*

Description

Plot a PST

Usage

```
## S4 method for signature 'PSTf,ANY'
plot(x, y=missing, max.level=NULL,
nodePar = list(), edgePar = list(),
axis=FALSE, xlab = NA, ylab = if (axis) { "L" } else {NA},
horiz = FALSE, xlim, ylim,
withlegend=TRUE, ltext=NULL, cex.legend=1,
use.layout=withlegend!=FALSE, legend.prop=NA, ...)
```

Arguments

x	A PST, that is an object of class `PSTf` as returned by the `pstree` or `prune` method.

y	not applicable
max.level	integer. The maximal depth for the display of the tree.
nodePar	list. A list of parameters for tuning the node representation. Possible parameters are • node.size. numeric. The size of the node, in fraction of a unit of the x axis (or y axis if horiz=TRUE). • gratio. The ratio between horizontal and vertical dimensions of the node. usefull if the horizontal and vertical dimensions of the plot are not equal. If not provided, it is estimated as a function of the number of leaves represented in the plot and the depth of the tree.
edgePar	list. A list of parameters for tuning the edges representation. Possible paramters are
axis	logical. If TRUE the axes are displayed on the plot.
xlab	character. Label for the x axis.
ylab	character. Label for the y axis representing the tree depth.
horiz	logical. If FALSE, the tree is represented vertically. The root node at depth L=0 is plotted on the top, and the nodes of maximal depth are plotted on the bottom of the plot. If TRUE, the tree is represented horizontally. The root node at depth L=0 is plotted on the right, and the nodes of maximal depth are plotted on the left of the plot.
xlim	numeric. Vector of length 2 giving the x limits for the plot. By default the limits are 1 .. number of terminal nodes (at max.level if specified). This may be usefull to facilitate comparison if several trees are plotted on the same figure.
ylim	numeric. Vector of length 2 giving the y limits for the plot. By default the limits are 0 .. max. depth of the tree (max.level if specified). This may be usefull to facilitate comparison if several trees are plotted on the same figure.
withlegend	defines if and where the legend of the state colors is plotted. The default value TRUE sets the position of the legend automatically. Other possible value is "right".
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.
...	arguments to be passed to the plot function or graphical parameters

Details

The function for graphical representation of a PST uses is recursive. The main argument of the function is a tree represented as a nested list (an object of class PSTr). See also *Gabadinho 2016.*

Author(s)

Alexis Gabadinho, based on code from plot.dendrogram

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

Examples

```
data(s1)
s1 <- seqdef(s1)
S1 <- pstree(s1, L=3)
plot(S1)
plot(S1, horiz=TRUE)
plot(S1, nodePar=list(node.type="path", lab.type="prob", lab.pos=1, lab.offset=2, lab.cex=0.7),
edgePar=list(type="triangle"), withlegend=FALSE)
```

---

pmine *PST based pattern mining*

Description

Mine for (sub)sequences satisfying user defined criteria in a state sequence object

Usage

```
## S4 method for signature 'PSTf,stslist'
pmine(object, data, l, pmin=0, pmax=1, prefix, lag, average=FALSE,
output="sequences", with.prefix=TRUE, sorted=TRUE, decreasing=TRUE, score.norm=FALSE)
```

Arguments

object	A fitted PST, that is an object of class PSTf as returned by the pstree or prune method.
data	A sequence object of class 'stslist' as defined with the seqdef function of the TraMineR library.
l	integer. Length of the subsequence to search for.
pmin	numeric. (Sub)-sequences having average or per state probability greater or equal than pmin are selected. Default to 1, meaning no lower threshold for the probability.

pmax	numeric. (Sub)-sequences having average or per state probability less or equal than pmax are selected. Default to 1, meaning no upper threshold for the probability.
prefix	character. Subsequences are searched in sequences starting with 'prefix', where 'prefix' is a string representing a subsequence with states separated by '-'. This option can be used to search for -most- likely patterns in sequences starting with 'prefix'.
lag	integer. The lag first states in the sequence are omitted. If prefix is
average	logical. If TRUE, the pmin or pmax probability is supposed to be the average state probability in the (sub)sequence. If FALSE (sub)sequences having every state probability less than pmax or greater than pmin are selected.
output	character. If output='sequences' the whole sequence(s) where the user defined criteria is satisfied are returned. If output='patterns' only the (sub)sequences satisfying the user defined criteria are returned.
with.prefix	logical. If 'output=patterns', should the patterns in the output be preceeded by their prefix, that is by the whole sub-sequence preceding the pattern.
sorted	logical. If 'sorted=TRUE', selected patterns or sequences are sorted according to their score, i.e., their average probability.
decreasing	logical. If 'sorted=TRUE', should sort order be decreasing or increasing ?
score.norm	logical. If TRUE, the score attached to each selected pattern or (sub)-sequence (the weights in the returned sequence object) is the average per state probability, and is thus normalized by the length of the pattern. If FALSE, the score is the whole (sub)-sequence probability.

Details

The likelihood $P^S(x)$ of a whole sequence x is computed from the state probabilities at each position in the sequence. However, the likelihood of the first states is usually lower than at higher position due to a reduced memory available for prediction. A sequence may not appear as very likely if its first state has a low relative frequency, even if the model predicts high probabilities for the states at higher positions.

The pmine function allows for advanced pattern mining with user defined parameters. It is controlled by the lag and pmin arguments. For example, by setting lag=2 and pmin=0.40 (example 1), we select all sequences with average (the geometric mean is used) state probability from position $lag + 1, \ldots, \ell$ above pmin. Instead of considering the average state probability at positions $lag + 1, \ldots, \ell$, it is also possible to select frequent patterns that do not contain any state with probability below the threshold. This prevents from selecting sequences having many states with high probability but one ore several states with a low probability.

It is also possible to mine the sequence data for frequent patterns of length $\ell_j < \ell$, regardless of the position in the sequence where they occur. By using the output="patterns" argument, the pmine function returns the patterns (as a sequence object) instead of the whole set of distinct sequences containing the patterns. Since the probability of a pattern can be different depending on the context (previous states) the returned subsequences also contain the context preceding the pattern. For more details, see *Gabadinho 2016*.

Value

A state sequence object, that is an object of class `stslist`, where weights are the probability score of (sub)sequences.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

See Also

`cmine` for context mining

Examples

```
## activity calendar for year 2000
## from the Swiss Household Panel
## see ?actcal
data(actcal)

## selecting individuals aged 20 to 59
actcal <- actcal[actcal$age00>=20 & actcal$age00 <60,]

## defining a sequence object
actcal.lab <- c("> 37 hours", "19-36 hours", "1-18 hours", "no work")
actcal.seq <- seqdef(actcal,13:24,labels=actcal.lab)

## building a PST
actcal.pst <- pstree(actcal.seq, nmin=2, ymin=0.001)

## pruning
## Cut-offs for 5% and 1% (see ?prune)
C99 <- qchisq(0.99,4-1)/2
actcal.pst.C99 <- prune(actcal.pst, gain="G2", C=C99)

## example 1
pmine(actcal.pst.C99, actcal.seq, pmin=0.4, lag=2)

## example 2: patterns of length 6 having p>=0.6
pmine(actcal.pst.C99, actcal.seq, pmin=0.6, l=6)
```

ppplot *Plotting a branch of a probabilistic suffix tree*

Description

The ppplot function displays the probability distributions of a node and all its parent nodes (suffixes) in the tree. IF the name of a gain function and a vector of pruning cutoffs are provided, the graphic will display the outcomes of the gain function, i.e., whether a node represents an information gain relative to its parent.

Usage

```
## S4 method for signature 'PSTf'
ppplot(object, path, gain, C, cex.plot = 1, nsize = 0.3, nlab=TRUE,
psize = nsize/2, pruned.col = "red", div.col = "green", ...)
```

Arguments

object	a probabilistic suffix tree, i.e., an object of class "PSTf" as returned by the pstree, prune or tune function.
path	character. Either a character string representing the node label (i.e., the context) where symbols are separated by '-', or a vector where each element is a symbol. See example.
gain	character or function. Gain function, see prune.
C	numeric. Value of the cutoff used by the gain function, see prune.
cex.plot	numeric. 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.
nsize	numeric. Size of the circles representing the nodes.
nlab	logical. Should the node label be displayed inside the circle?
psize	numeric. Size of the circles representing the outcome of the gain function.
pruned.col	character. Color used to represent a terminal node which provides no information gain relative to its parent.
div.col	character. Color used to represent an internal node which provides information gain relative to its parent.
...	additional parameters to be passed to the plot function.

Details

For more details, see *Gabadinho 2016*.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

See Also

cplot, prune

Examples

```
data(s1)
s1.seq <- seqdef(s1)
S1 <- pstree(s1.seq, L=5, ymin=0.001)
ppplot(S1, "a-a-b-b-a", gain="G1", C=c(1.1, 1.2))
```

---

pqplot *Prediction quality plot*

Description

Plot the predicted probability of each state in a sequence

Usage

```
## S4 method for signature 'PSTf,stslist'
pqplot(object, data, cdata, L, stcol, plotseq=FALSE,
ptype="b", cex.plot=1, space=0,
measure="prob", pqmax, seqscale, ...)
```

Arguments

object	a probabilistic suffix tree, i.e., an object of class "PSTf" as returned by the pstree, prune or tune function.
data	a sequence object, i.e., an object of class 'stslist' as created by TraMineR seqdef function, either subsetted with the index of the sequence to predict or containing one sequence.
cdata	Not implemented yet.
L	integer. Maximal context length for sequence prediction. This is the same as pruning the PST by removing all nodes of depth<L before prediction.
stcol	character. Color to use to plot the prediction qualities.
plotseq	logical. If TRUE, the sequence is displayed separately, and the prediction plot is plotted above.
ptype	character. Type of plot, either 'b' for barplot or 'l' for line.
cex.plot	numeric. 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.

space	numeric. Space separating each state in the plot.
measure	character. Measure used for prediction quality. Either 'prob' or 'logloss'.$
pqmax	numeric. Maximum coordinate for the prediction quality plot, i.e. the max of the y axis.
seqscale	numeric. If plotseq=TRUE, width of the bar representing the sequence as a proportion of the y axis range.
...	optional graphical parameters to be passed to the plot function.

Details

The pqplot() function displays either the predicted probabilities or the log-loss for each position of a single sequence as a series of barplots. For more details, see *Gabadinho 2016.*

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016) Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), 1-39.

Examples

```
data(s1)
s1 <- seqdef(s1)
S1 <- pstree(s1, L=3)

z <- seqdef("a-b-a-a-b")
pqplot(S1, z)
pqplot(S1, z, measure="logloss", plotseq=TRUE)
```

---

predict *Compute the probability of categorical sequences using a probabilistic suffix tree*

Description

Compute the probability (likelihood) of categorical sequences using a Probabilistic Suffix Tree

Usage

```
## S4 method for signature 'PSTf'
predict(object, data, cdata, group, L=NULL, p1=NULL, output="prob", decomp=FALSE, base=2)
```

Arguments

object	a probabilistic suffix tree, i.e., an object of class "PSTf" as returned by the pstree, prune or tune function.
data	a sequence object, i.e., an object of class 'stslist' as created by TraMineR seqdef function, containing the sequences to predict.
cdata	not implemented yet.
group	if object is a segmented PST, providing a vector of group membership so that each sequence probability will be predicted with the conditional probability distributions for the group it belongs to. If object is a segmented PST and group is not provided, each sequence will be predicted by each of the submodel, and the output will be a matrix with nbgroup columns, where nbgroup is the number of segments in the PST.
L	integer. Maximal context length for sequence prediction. This is the same as pruning the PST by removing all nodes of depth<L before prediction.
p1	vector. A probability distribution for the first position in the sequence that will be used instead of the root node of the tree.
output	character. One of 'prob', 'logloss', 'SIMn' or 'SIMo'. See details.
decomp	logical. If TRUE the predicted probability for each state in the sequence(s) is returned instead of the whole sequence probability.
base	integer. Base for the logarithm if a logarithm is used in the used prediction measure.

Details

A probabilistic suffix tree (PST) allows to compute the likelihood of any sequence built on the alphabet of the learning sample. This feature is called sequence prediction. The likelihood of the sequence a-b-a-a-b given a PST S1 fitted to the example sequence s1 (see example) is

$$P^{S1}(abaab) = P^{S1}(a) \times P^{S1}(b|a) \times P^{S1}(a|ab) \times P^{S1}(a|aba) \times P^{S1}(b|abaa)$$

The probability of each of the state is retrieved from the PST. To get for example P(a|a-b-a), the tree is scanned for the node labelled with the string a-b-a, and if this node does not exist, it is scanned for the node labelled with the longest suffix of this string, that is b-a, and so on. The node a-b-a is not found in the tree (it has been removed during the pruning stage), and the longest suffix of a-b-a found is b-a. The probability P(a|b-a) is then used instead of P(a|a-b-a).

The sequence likelihood is returned by the predict function. By setting decomp=TRUE the output is a matrix containing the probability of each of the symbol composing the sequence. The score $P^S(x)$ of a sequence x represents the probability that the VLMC model stored by the PST S generates x. It can be turned into a more readable prediction quality measure such as the *average log-loss*

$$logloss(S,x) = -\frac{1}{\ell}\sum_{i=1}^{\ell}\log_2 P^S(x_i|x_1,\ldots,x_{i-1}) = -\frac{1}{\ell}\log_2 P^S(x)$$

by using 'output=logloss'. The returned value is the average log-loss of each state in the sequence, which allows to compare the prediction for sequences of unequal lengths. The average

log-loss can be interpreted as a residual, that is the distance between the prediction of a sequence by a PST S and the perfect prediction $P(x) = 1$ yielding $logloss(P^S, x) = 0$. The lower the value of $logloss(P^S, s)$ the better the sequence is predicted. For more details, see *Gabadinho 2016*.

Value

Either a vector of sequence probabilities (decomp=FALSE) or a matrix (if decomp=FALSE) containing for each sequence (row) the probability of each state in columns.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016) Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), 1-39.

Examples

```
data(s1)
s1 <- seqdef(s1)

S1 <- pstree(s1, L=3, nmin=2, ymin=0.001)
S1 <- prune(S1, gain="G1", C=1.20, delete=FALSE)

predict(S1, s1, decomp=TRUE)
predict(S1, s1)
```

---

print *Print method for objects of class* PSTf *and* PSTr

Description

Display a probabilistic suffix tree

Usage

```
## S4 method for signature 'PSTr'
print(x, max.level = NULL, digits = 1, give.attr = FALSE,
    nest.lev = 0, indent.str = "", stem = "--")
```

Arguments

x	A PST, that is an object of class PSTf as returned by the pstree or prune method.
max.level	integer. The maximal depth for the display of the tree.
digits	integer specifying the precision for printing.

give.attr	logical. If TRUE the attributes of each node (an object of class PSTr) are displayed.
nest.lev	integer. Parameter used internally by the function.
indent.str	character. String used to indent each line when displaying the tree. Default to ".
stem	character. String used to display the stems. Default to '–'.

Methods

signature(x = "ANY")

signature(x = "PSTf")

signature(x = "PSTr")

---

prune *Prune a probabilistic suffix tree*

Description

Prune a PST, using either a gain function, a maximal depth or a list of nodes to keep or remove. Optionally, nodes are not removed from the tree but tagged as deleted, helping to visualize the pruning process.

Usage

```
## S4 method for signature 'PSTf'
prune(object, nmin, L, gain, C, keep, drop, state, delete = TRUE, lik =TRUE)
```

Arguments

object	a probabilistic suffix tree, i.e., an object of class "PSTf" as returned by the pstree, prune or tune function.
nmin	integer. All strings having counts less than nmin are removed.
L	integer. If specified the the tree is cut at depth L., that is all nodes with depth > L are removed.
gain	character. Function for measuring information gain. See details.
C	numeric. Cutoff value to use with the gain function
keep	character. A vector of character strings containing the names of the nodes to keep in the tree. All nodes that are not a suffix of contexts in keep are removed from the tree.
drop	character. A vector of character strings containing the names of the nodes to remove from the tree. All nodes that are a suffix of contexts in drop are removed from the tree as weel.
state	character. All nodes corresponding to contexts which include state are pruned.

`delete` Logical. If FALSE, the pruned nodes are not removed from the tree but tagged as pruned=FALSE, so that when plotting the pruned tree these nodes wil appear surrounded with red (can be set to another color) lines.

`lik` Logical. If TRUE, the log-likelihood of the pruned model, i.e. the likelihood of the training sequences given the model, is computed and stored in the 'logLik' slot of the PST. Setting to FALSE will spare the time required to compute the likelihood.

Details

The initial tree returned by the `pstree` function may yield an overly complex model containing all contexts of maximal length L and frequency $N(c) \geq nmin$ found in the learning sample. The pruning stage potentially reduces the number of nodes in the tree, and thus the model complexity. It compares the conditional probabilities associated to a node labelled by a subsequence $c = c_1, c_2, \ldots, c_k$ to the conditional probabilities of its parent node labelled by the longest suffix of c, $suf(c) = c_2, \ldots, c_k$. The general idea is to remove a node if it does not contribute additional information with respect to its parent in predicting the next symbol, that is if $\hat{P}(\sigma|c)$ is not *significantly* different from $\hat{P}(\sigma|suf(c))$ for all $\sigma \in A$.

The pruning procedure starts from the terminal nodes and is applied recursively until all terminal nodes remaining in the tree represent an information gain relative to their parent. A gain function, whose outcome will determine the pruning decision, is used to compare the two probability distributions. The gain function is driven by a cut-off, and different values of this parameter will yield more or less complex trees. A method for selecting the pruning cut-off is described in the `tune` help page.

A first implemented gain function, which is used by the *Learn-PSA* algorithm, is based on the ratio between $\hat{P}(\sigma|c)$ and $hatP(\sigma|suf(c))$ for each $\sigma \in A$. A node represents an information gain if for any symbol $\sigma \in A$ the ratio is greater than the cut-off C or lower than $1/C$, that is if

$$G_1(c) = \sum_{\sigma \in A} 1 \left[\frac{\hat{P}(\sigma|c)}{\hat{P}(\sigma|suf(c))} \geq C \ \cup \ \frac{\hat{P}(\sigma|c)}{\hat{P}(\sigma|suf(c))} \leq \frac{1}{C} \right] \geq 1$$

where C is a user defined cut-off value. Nodes that do not satisfy the above condition are pruned. For $C = 1$ no node is removed since even a node having a next probability distribution similar to the one of its parent does not satisfy the pruning condition.

The *context* algorithm uses another gain function, namely

$$G_2(c) = \sum_{\sigma \in A} \hat{P}(\sigma|c) \log \left(\frac{\hat{P}(\sigma|c)}{\hat{P}(\sigma|suf(c))} \right) N(c) > C$$

where c is the context labelling the terminal node, $N(c)$ is the number of occurrences of c in the data. The cutoff C is specified on the scale of χ^2-quantiles *Maechler-2004*

$$C = C(\alpha) = \frac{1}{2} qchisq(1 - \alpha, v), v = |A| - 1$$

where $qchisq(p = 1 - \alpha, v)$ is the quantile function of a χ^2 distribution with v degrees of freedom. The cutoff C is a threshold for the difference of deviances between a tree S^1 and its subtree S^2 obtained by pruning the terminal node c. Typical values for α are 5% and 1%, yielding $p = 0.95$ and $p = 0.99$ respectively. For more details, see *Gabadinho 2016*.

Value

A probabilistic suffix tree, i.e., an object of class PSTf.

Author(s)

Alexis Gabadinho

References

Bejerano, G. & Yona, G. (2001). Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, 17, pp. 23-43.

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

Maechler, M. & Buehlmann, P. (2004). Variable Length Markov Chains: Methodology, Computing, and Software *Journal of Computational and Graphical Statistics*, 13, pp. 435-455.

Ron, D.; Singer, Y. & Tishby, N. (1996). The power of amnesia: Learning probabilistic automata with variable memory length *Machine Learning*, 25, pp. 117-149.

See Also

tune, ppplot

Examples

```
data(s1)
s1.seq <- seqdef(s1)
S1 <- pstree(s1.seq, L=3, nmin=2, ymin=0.001)

## --
S1.p1 <- prune(S1, gain="G1", C=1.20, delete=FALSE)
summary(S1.p1)
plot(S1.p1)

## --
C95 <- qchisq(0.95,1)/2
S1.p2 <- prune(S1, gain="G2", C=C95, delete=FALSE)
plot(S1.p2)
```

PSTf-class *Flat representation of a probabilistic suffix tree*

Description

The class "PSTf" is the flat representation of a probabilistic suffix tree (PST) storing a variable length Markov chain model. The flat representation is a list where each element corresponds to a given depth. It is the prefered representation and is used by all functions for model fitting and sequence analysis with PST. The nested representation "PSTr" is used only for printing and plotting PSTs.

Objects from the Class

Objects of class "PSTf" are returned by the pstree, prune and tune function.

Slots

.Data: Object of class "list", a list where each element corresponds to one level of the tree and is itself a list of nodes, i.e., objects of class "PSTr".

data: Object of class "stslist". The learning sample to which the PST is fitted, i.e., a sequence object created with the seqdef function.

cdata: Object of class "stslist"

alphabet: Object of class "character". Alphabet on which the sequences, and the PST are built.

labels: Object of class "character" containing the long state labels.

cpal: Object of class "character". Color palette used to represent each state of the alphabet.

segmented: Object of class "logical" indicating whether the tree is segmented. See pstree.

group: Object of class "factor" containing the group membership for each sequence in data.

call: Object of class "call".

logLik: Object of class "numeric", containing the log-likelihood of the VLMC model represented by the PST.

Extends

Class "list", from data part. Class "vector", by class "list", distance 2.

Methods

cmine signature(object = "PSTf"): context mining, see cmine,PSTf-method.

cplot signature(object = "PSTf"): plot single nodes of a PST, see cplot,PSTf-method.

generate signature(object = "PSTf"): generate artificial sequences, see generate,PSTf-method.

impute signature(object = "PSTf", data = "stslist"): impute missing values in sequence data, seeimpute,PSTf,stslist-method.

logLik signature(object = "PSTf"): extract log-likelihood of the VLMC model represented by a PST, see logLik,PSTf-method.

nobs signature(object = "PSTf"): number of observations (symbols) in the learning sample to which a VLMC model is fitted, see nobs,PSTf-method.

nodenames signature(object = "PSTf"): retrieve the node labels of a PST, see see nodenames,PSTf-method.

pdist signature(x = "PSTf", y = "PSTf"): compute probabilistic divergence between two PSTs, see pdist,PSTf,PSTf-method.

plot signature(x = "PSTf", y = "ANY"): plot a PST, see plot,PSTf,ANY-method.

pmine signature(object = "PSTf", data = "stslist"): pattern mining, see see pmine,PSTf,stslist-method.

ppplot signature(object = "PSTf"): plotting a branch of a PST, see ppplot,PSTf-method.

pqplot signature(object = "PSTf", data = "stslist"): plot the predicted probability of each state in a sequence, see pqplot,PSTf,stslist-method.

predict `signature(object = "PSTf")`: predict the likelihood of sequences, see `predict,PSTf-method`.

print `signature(x = "PSTf")`: print a PST, see `print,PSTf-method`.

prune `signature(object = "PSTf")`: prune a PST, see `prune,PSTf-method`.

query `signature(object = "PSTf")`: retrieve counts or next symbol probability distribution from a node in a Probabilistic Suffix Tree, see `query,PSTf-method`.

subtree `signature(object = "PSTf")`: extract a subtree from a segmented PST, see `subtree,PSTf-method`.

summary `signature(object = "PSTf")`: see `summary,PSTf-method`.

tune `signature(object = "PSTf")`: AIC, AICc and BIC based model selection, see `tune,PSTf-method`.

Author(s)

Alexis Gabadinho

See Also

`PSTr`

Examples

```
showClass("PSTf")
```

PSTr-class *Nested representation of a probabilistic suffix tree*

Description

An object of class `"PSTr"` is a node of a probabilistic suffix tree (PST). The slot `prob` contains one or several probability distributions (if the PST is segmented) and the slot `counts` contains the empirical - possibly weighted - counts from which the probabilities are computed. The slot `leaf` indicates whether the node (segment) is a terminal node (segment). The 'flat' representation of a PST is an object of class `"PSTf"`), that is a list that contains one element for each level of the tree. Each element of the list is itself a list whose elements are nodes, that is objects of class `PSTr`. The 'nested' representation of a probabilistic suffix tree (PST) is a nested list whose elements are children nodes of class `"PSTr"`. This representation is used for printing and plotting PST, in which case the flat representation of a PST, i.e., an object of class `"PSTf"` is turned into an object of class `"PSTr"` by using the `as` function.

Objects from the Class

Objects are created when calling the `pstree` function.

Slots

`.Data`: Object of class `"list"`. In the nested representation of a PST, the elements of the list are the children nodes. Otherwise the list is empty.

`alphabet`: Object of class `"character"`. Alphabet on which the sequences, and the PST are built. This slot is non-empty only for the root node of the nested representation of a PST.

`labels`: Object of class `"character"` containing the long state labels. This slot is non-empty only for the root node of the nested representation of a PST.

`cpal`: Object of class `"character"`. Color palette used to represent each state of the alphabet. This slot is non-empty only for the root node of the nested representation of a PST.

`index`: Object of class `"matrix"`. When the PST is segmented, indicates the id of the segment corresponding to each group.

`counts`: Object of class `"matrix"`. The counts to which the probability distributions are computed.

`n`: Object of class `"matrix"`. The number of occurrences of the context in the learning sample, see `cprob`.

`prob`: Object of class `"matrix"`. The probability distributions computed from the counts.

`path`: Object of class `"character"`. The node label, i.e. the context which is the path from the node to the root node of the tree.

`order`: Object of class `"integer"`. The depth of the node in the tree, i.e., the order of the probability distribution(s) stored in the node.

`leaf`: Object of class `"matrix"`. Indicates whether the node (segment) is a terminal node (segment).

`pruned`: Object of class `"matrix"`. If the PST was pruned with the `delete=FALSE` option, indicates whether the node (segment) is actually pruned. See `prune`.

Extends

Class `"list"`, from data part. Class `"vector"`, by class "list", distance 2.

Methods

[[`signature(x = "PSTr")`: extract sub-branches of a nested representation of a PST.

plot `signature(x = "PSTr", y = "ANY")`: plot a PST, see `plot,PSTr,ANY-method`.

print `signature(x = "PSTr")`: print a PST, see `print,PSTr-method`.

summary `signature(object = "PSTr")`: see `summary,PSTr-method`.

Author(s)

Alexis Gabadinho

See Also

`PSTf`

Examples

```
showClass("PSTr")
```

pstree *Build a probabilistic suffix tree*

Description

Build a probabilistic suffix tree that stores a variable length Markov chain (VLMC) model

Usage

```
## S4 method for signature 'stslist'
pstree(object, group, L, cdata=NULL, stationary=TRUE,
nmin = 1, ymin=NULL, weighted = TRUE, with.missing = FALSE, lik = TRUE)
```

Arguments

object	a sequence object, i.e., an object of class 'stslist' as created by TraMineR seqdef function.
group	a vector giving the group membership for each observation in x. If specified, a segmented PST is produced containing one PST for each group.
cdata	Not implemented yet.
stationary	Not implemented yet.
L	Integer. Maximal depth of the PST. Default to maximum length of the sequence(s) in object minus 1.
nmin	Integer. Minimum number of occurences of a string to add it in the tree
ymin	Numeric. Smoothing parameter for conditional probabilities, assuring that no symbol, and hence no sequence, is predicted to have a null probability. The parameter $ymin$ sets a lower bound for a symbol's probability.
weighted	Logical. If TRUE, weights attached to the sequence object are used in the estimation of probabilities.
with.missing	Logical. If TRUE, the missing state is added to the alphabet
lik	Logical. If TRUE, the log-likelihood of the model, i.e. the likelihood of the training sequences given the model, is computed and stored in the 'logLik' slot of the PST. Setting to FALSE will spare the time required to compute the likelihood.

Details

A probabilistic suffix tree (PST) is built from a learning sample of $n,\ n \geq 1$ sequences by successively adding nodes labelled with subsequences (contexts) c of length $L,\ 0 \leq L \leq L_{max}$ found in the data. When the value L_{max} is not defined by the user it is set to its theorectical maximum $\ell - 1$ where ℓ is the maximum sequence length in the learning sample. The nmin argument specifies the minimum frequency of a subsequence required to add it to te tree.
Each node of the tree is labelled with a context c and stores the next symbol empirical probability distribution $\hat{P}(\sigma|c),\ \sigma \in A$, where A is an alphabet of finite size. The root node labelled with

the empty string e stores the $0th$ order probability $\hat{P}(\sigma),\ \sigma \in A$ of oberving each symbol of the alphabet in the whole learning sample.
The building algorithm calls the cprob function which returns the empirical next symbol counts observed after each context c and computes the corresponding empirical probability distribution. Each node in the tree is connected to its longest suffix, where the longest suffix of a string $c = c_1, c_2, \ldots, c_k$ of length k is $suffix(c) = c_2, \ldots, c_k$.
Once an initial PST is built it can be pruned to reduce its complexity by removing nodes that do not provide significant information (see prune). A model selection procedure based on information criteria is also available (see tune). For more details, see *Gabadinho 2016*.

Value

An object of class "PSTf".

Author(s)

Alexis Gabadinho

References

Bejerano, G. & Yona, G. (2001) Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics* 17, 23-43.

Gabadinho, A. & Ritschard, G. (2016) Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software* **72**(3), 1-39.

Maechler, M. & Buehlmann, P. (2004) Variable Length Markov Chains: Methodology, Computing, and Software. *Journal of Computational and Graphical Statistics* 13, pp. 435-455.

Ron, D.; Singer, Y. & Tishby, N. (1996) The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning* 25, 117-149.

See Also

prune, tune

Examples

```
## Build a PST on one single sequence
data(s1)
s1.seq <- seqdef(s1)
s1.seq
S1 <- pstree(s1.seq, L = 3)
print(S1, digits = 3)
S1
```

query *Retrieve counts or next symbol probability distribution*

Description

Retrieve counts or next symbol probability distribution from a node of a probabilistic suffix tree

Usage

```
## S4 method for signature 'PSTf'
query(object, context, state, output = "prob", exact = FALSE)
```

Arguments

object	A probabilistic suffix tree, i.e an object of class "PSTf") as returned by the pstree, prune or tune function.
context	Character. The string labelling the node to retrieve. States must be separated by '-' as for example in 'a-a-b'. If the node labelled with this string does not exist in the tree, the node labelled with the longest suffix is searched for, and so on until an existing node is found.
state	character. If specified the probability of the specified state is returned instead of the whole distribution.
output	character. If output="prob" the probability distribution (or a single symbol distribution if state is specified) is returned. If output="counts" the counts on which the probability distribution is calculated are returned. If output="all" the node itself is returned, that is an object of class PSTr.
exact	logical. If TRUE, the information is returned only if the node labelled with context is present in the tree. That is, the longest suffix of context is not searched for if context is not in the tree.

Details

The PST is searched for the node labelled with context. If exact=FALSE, when the node does not exist the PST is searched for the longest suffix of context, and so on until a node corresponding to a suffix of context is found or the root node is reached. For more details, see *Gabadinho 2016*.

Value

An object of class cprobd, with available round method.

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

See Also

cplot, ppplot

Examples

```
data(s1)
s1 <- seqdef(s1)
S1 <- pstree(s1, L=3)
## Retrieving from the node labelled 'a-a-a'
query(S1, "a-a-a")

## The node 'a-b-b-a' is not presetnin the tree, and the next symbol
## probability is retrieved from the node labelled 'b-b-a' (the longest
## suffix
query(S1, "a-b-b-a")
```

---

s1 *Example sequence data set*

Description

Example data set containing one single sequence

Usage

```
data(s1)
```

Format

A character string representing a sequence of 27 symbols separated with '-'.

Details

A sequence object can be created with the dedicated TraMineR seqdef function. State sequence objects are the main argument for the pstree method that creates probabilistic suffix trees. See example below.

Examples

```
## Loading the data
data(s1)

## Creating a state sequence object
s1.seq <- seqdef(s1)

## Building and plotting a PST
S1 <- pstree(s1.seq, L = 3)
plot(S1)
```

SRH *Longitudinal data on self rated health*

Description

Longitudinal data on self rated health from waves 1-11 of the Swiss household panel

Usage

```
data(SRH)
```

Format

SRH is a data frame with 2612 observations on the following 15 variables.

`idpers` personal identification number

`sex` a factor with levels `man` `woman`

`birthy` birth year of the respondent

`wp09lp1s` longitudinal weight

`p99c01` **...** `p09c01` factors with levels:
`very well`; `well`; `so`, `so (average)`; `not very well`; `not well at all`

`SRH.seq` is a `TraMineR` sequence object created from the `SRH` data frame using the code in `example`. States are coded as follows:

G1	(very well)
G2	(well)
M	(so, so (average))
B2	(not very well)
B1	(not well at all)

Details

Respondant's self rated health is collected at each yearly wave of the SHP with the following question: *How do you feel right now?*. Possible answers are: very well; well; so, so (average), not very well and not well at all. The sequences are made of an individual's responses over 11 yearly waves of the SHP, starting with wave 1 in 1999. Variable p99c01 contains the self rated health at wave 1, p00c01 contains the self rated health at wave 2, etc... Note that sequences may contain missing values due to wave or item non response.

Source

Swiss Household Panel: https://forscenter.ch/projects/swiss-household-panel/

Examples

```
## Preparing a sequence object with the SRH data set
data(SRH)

## Long state labels
state.list <- levels(SRH$p99c01)

## Sequential color palette
mycol5 <- rev(brewer.pal(5, "RdYlGn"))

## Creating the sequence object
SRH.seq <- seqdef(SRH, 5:15, alphabet=state.list,
states=c("G1", "G2", "M", "B2", "B1"), labels=state.list,
weights=SRH$wp09lp1s, right=NA, cpal=mycol5)
names(SRH.seq) <- 1999:2009
```

---

subtree *Extract a subtree from a segmented PST*

Description

Extract a subtree from a segmented PST

Usage

```
## S4 method for signature 'PSTf'
subtree(object, group=NULL, position=NULL)
```

Arguments

object	A segmented probabilistic suffix tree, i.e an object of class "PSTf") as returned by the pstree, prune or tune function.
group	integer. Segment of the PST
position	Not implemented yet.

Details

See also *Gabadinho 2016.*

Author(s)

Alexis Gabadinho

References

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

Examples

```
## activity calendar for year 2000
## from the Swiss Household Panel
## see ?actcal
data(actcal)

## selecting individuals aged 20 to 59
actcal <- actcal[actcal$age00>=20 & actcal$age00 <60,]

## defining a sequence object
actcal.lab <- c("> 37 hours", "19-36 hours", "1-18 hours", "no work")
actcal.seq <- seqdef(actcal,13:24,labels=actcal.lab)

## building a PST segmented by age group
gage10 <- cut(actcal$age00, c(20,30,40,50,60), right=FALSE,
labels=c("20-29","30-39", "40-49", "50-59"))

actcal.pstg <- pstree(actcal.seq, nmin=2, ymin=0.001, group=gage10)

## pruning
C99 <- qchisq(0.99,4-1)/2
actcal.pstg.opt <- prune(actcal.pstg, gain="G2", C=C99)

## extracting PST for age group 20-39 and 30-39
g1.pst <- subtree(actcal.pstg.opt, group=1)
g2.pst <- subtree(actcal.pstg.opt, group=2)

## plotting the two PST
par(mfrow=c(1,2))
plot(g1.pst, withlegend=FALSE, max.level=4, main="20-29")
plot(g2.pst, withlegend=FALSE, max.level=4, main="30-39")
```

summary-methods *Summary of variable length Markov chain model*

Description

Summary of a variable length Markov chain model stored in a probabilistic suffix tree.

Usage

```
## S4 method for signature 'PSTf'
summary(object, max.level)
```

Arguments

object	A PST, that is an object of class PSTf as returned by the pstree or prune method.
max.level	integer. If specified, the summary is computed for the max.level levels of the tree only.

Value

An object of class PST.summary with following attributes:

alphabet list of symbols in the alphabet

labels long labels for symbols in the alphabet

cpal color palette used to represent each state of the alphabet

ns number of symbols in the data to which the model was fitted

depth maximum depth (order) of the tree

nodes number of internal nodes in the PST

leaves number of leaves in the PST

freepar number of free parameters in the mode, i.e., (nodes+leaves)*(|A|-1) where |A| is the size of the alphabet

A show method is available for displaying objects of class PST.summary.

Author(s)

Alexis Gabadinho

Examples

```
data(s1)
s1.seq <- seqdef(s1)
S1 <- pstree(s1.seq, L=3)
summary(S1)
summary(S1, max.level=2)
```

tune *AIC, AICc or BIC based model selection*

Description

Prune a probabilistic suffix tree with a series of cut-offs and select the model having the lowest value of the selected information criterion. Available information criterion are Akaike information criterion (AIC), AIC with a correction for finite sample sizes (AICc) and Bayesian information criterion (BIC).

Usage

```
## S4 method for signature 'PSTf'
tune(object, gain="G2", C, criterion = "AIC", output = "PST")
```

Arguments

`object`	a probabilistic suffix tree, i.e., an object of class `"PSTf"` as returned by the `pstree`, `prune` or `tune` function.
`gain`	character. The gain function used for pruning decisions. See `prune` for details.
`C`	numeric. A vector of cutoff values. See `prune` for details.
`criterion`	The criterion used to select the model, either AIC, AICc or BIC. AICc should be used when the ratio between the number of observations and the number of estimated parameters is low, which is often the case with VLMC models. *Burnham et al 2004* suggest to use AICc instead of AIC when the ratio is lower than 40.
`output`	If `output='PST'` the PST (an object of class `"PSTr"`) having the lowest AIC, AICc or BIC value. If `output='stats'`, a table with the statistics for each model obtained by pruning `object` with the cut-offs in `C`.

Details

The `tune` function selects among a series of PST pruned with different values of the C cutoff the model having the lowest AIC or AIC_c value. The function can return either the selected PST or a data frame containing the statistics for each model. For more details, see *Gabadinho 2016*.

Value

If `output="PST"` a PST that is an object of class `PSTf`. If `output="stats"` a matrix with the results of the tuning procedure.
The selected model is tagged with `***`, while models with $IC < min(IC) + 2$ are tagged with `**`, and models with $IC < min(IC) + 10$ are tagged with `**`.

Author(s)

Alexis Gabadinho

References

Burnham, K. P. & Anderson, D. R. (2004). Multimodel Inference *Sociological Methods & Research*, 33, pp. 261-304.

Gabadinho, A. & Ritschard, G. (2016). Analyzing State Sequences with Probabilistic Suffix Trees: The PST R Package. *Journal of Statistical Software*, **72**(3), pp. 1-39.

See Also

prune

Examples

```
## activity calendar for year 2000
## from the Swiss Household Panel
## see ?actcal
data(actcal)

## selecting individuals aged 20 to 59
actcal <- actcal[actcal$age00>=20 & actcal$age00 <60,]

## defining a sequence object
actcal.lab <- c("> 37 hours", "19-36 hours", "1-18 hours", "no work")
actcal.seq <- seqdef(actcal,13:24,labels=actcal.lab)

## building a PST
actcal.pst <- pstree(actcal.seq, nmin=2, ymin=0.001)

## Cut-offs for 5% and 1% (see ?prune)
C95 <- qchisq(0.95,4-1)/2
C99 <- qchisq(0.99,4-1)/2

## selecting the optimal PST using AIC criterion
actcal.pst.opt <- tune(actcal.pst, gain="G2", C=c(C95,C99))

## plotting the tree
plot(actcal.pst.opt)
```

Index