Package 'hmm.discnp'

July 22, 2025

Version 3.0-9

Date 2022-09-26

Title Hidden Markov Models with Discrete Non-Parametric Observation Distributions

Author Rolf Turner

Maintainer Rolf Turner <r.turner@auckland.ac.nz>

Depends R (>= 2.10)

Imports nnet (>= 7.3.12)

Description Fits hidden Markov models with discrete non-parametric observation distributions to data sets. The observations may be univariate or bivariate. Simulates data from such models. Finds most probable underlying hidden states, the most probable sequences of such states, and the log likelihood of a collection of observations given the parameters of the model. Auxiliary predictors are accommodated in the univariate setting.

LazyData true

ByteCompile true

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-09-26 09:10:06 UTC

Contents

`anova.hmm.discnp` *Anova for hmm.discnp models*

Description

Performs a likelihood ratio test to compare two discrete non-parametric hidden Markov models.

Usage

```
## S3 method for class 'hmm.discnp'
anova(object, ...)
```

Arguments

`object`	An object of class "hmm.discnp" as returned by the function `hmm()`.
`...`	A second object of class "hmm.discnp". There must be only *one* such object.

Value

A list with entries

`stat`	The likelihood ratio statistic.
`df`	The degrees of freedom.
`pvalue`	The p-value of the test.

This list has an attribute "details" which is a vector consisting of the first and second log likelihoods and the associated numbers of parameters, in order of these numbers of parameters. (See **Warning**.)

Warning

Hidden Markov models can be numerically delicate and the fitting algorithm can converge to a local maximum of the likelihood surface which is not the global maximum. Thus it is entirely possible for the log likelihood of the model with the greater number of parameters to be *smaller* than that of the model with the lesser number of parameters.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

hmm()

Examples

```
xxx  <- with(SydColDisc,split(y,f=list(locn,depth)))
fit1 <- hmm(xxx,K=1,itmax=10)
fit2 <- hmm(xxx,K=2,itmax=10)
anova(fit1,fit2)
```

---

ccprSim *Simulated monocyte counts and psychosis symptoms.*

Description

Discretised values of monocyte counts, and ratings of level of psychosis simulated from a model fitted to a data set consisting of observations made on a number of patients from the Northern District Health Board system. The real data must be kept confidential due to ethics constraints.

Usage

```
data("ccprSim")
```

Format

The object ccprSim is a list of length 1258. Each entry of this list is to be considered to correspond to an individual subject. The entries consist of matrices having two columns named cellCount and psychosisRating. The number of rows of these matrices varies from entry to entry of the list (i.e. from subject to subject).

Most of the entries of these matrices are NA. The entries are temporally ordered and correspond to the number of weeks from the start of observation. Observations in the real data set were made only when the patient in question visted a physician and so weeks in which no visit was made resulted in an "observation" of NA. The object ccprSim was simulated in such a way as to imitate this characteristic. The fraction of missing observations in each variate (i.e. cellCount and psychosisRating is roughly commensurate with the corresponding fractions in the real data.

The values in the first column of each matrix (the `cellCount` column) consist of integers from 1 to 5 and are to be interpreted as indicators of cell counts in units of 10^9 cells per litre, discretised according to the following scale:

- $0.0 \leq c \leq 0.3 \leftrightarrow 1$
- $0.3 < c \leq 0.5 \leftrightarrow 2$
- $0.5 < c \leq 0.7 \leftrightarrow 3$
- $0.7 < c \leq 1.0 \leftrightarrow 4$
- $1.0 < c \leq 2.0 \leftrightarrow 5$

where c represents "count".

The values in the second column of each matrix (the `psychosisRating` column consist of integers from 0 to 4 and are to be interpreted as indicators of a physician's assessment of the level of pschosis of the patient. A value of 0 corresponds to "no symptoms"; a value of 4 corresponds to "severe".

The question of essential interest in respect of the real data was "Is there any association between the cell count values and the psychosis ratings?" More specifically it was "Can the level of psychosis be *predicted* from the cell counts?"

Source

The real data, on the basis of which these data were simulated, were supplied by Dr. Jonathan Williams, Northern District Health Board.

Examples

```
## Not run:  # Takes too long.
    fit <- hmm(ccprSim,K=2,indep=FALSE,itmax=5,verbose=TRUE)

## End(Not run)
```

---

cnvrtRho *Convert Rho between forms.*

Description

Converts a matrix specification of the emission probabilities (in which the probabilities are simply the entries of the matrix) to a data frame specification (in which the probabilities are a logistic-style function of the parameters) or vice versa.

Usage

```
cnvrtRho(Rho)
```

Arguments

Rho	A specification of the emission probabilities of a discrete valued hidden Markov model. It may be either a matrix of these probabilities, in which case it is converted to a three column data frame, or it may be a three column data frame, in which case it is converted to a matrix of probabilities. See hmm() for more details about the structure of Rho, in either form.

Details

The data frame specification of Rho allows for predictor variables x. If Rho is of the data frame form, and is designed to allow for predictor variables, then it will have more than three columns and cannot be converted to the matrix form. In such cases cnvrtRho will throw an error.

Value

A data frame if the argument Rho is a matrix, or a matrix if the argument Rho is a data frame.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

hmm()

Examples

```
Yval <- LETTERS[1:10]
Tpm  <- matrix(c(0.75,0.25,0.25,0.75),ncol=2,byrow=TRUE)
Rho  <- cbind(c(rep(1,5),rep(0,5)),c(rep(0,5),rep(1,5)))/5
rownames(Rho) <- Yval
newRho <- cnvrtRho(Rho)
oldRho <- cnvrtRho(newRho)
```

---

fitted.hmm.discnp *Fitted values of a discrete non-parametric hidden Markov model.*

Description

Calculates the fitted values of a discrete non-parametric hidden Markov model. If the data are numeric these are the conditional expectations of the observations, given the entire observation sequence (and the estimated parameters of the model). If the data are categorical (whence "expectations" make no sense) the "fitted values" are taken to be the probabilities of each of the possible values of the observations, at each time point.

Usage

```
## S3 method for class 'hmm.discnp'
fitted(object, warn=TRUE, drop=TRUE, ...)
```

Arguments

object	An object of class hmm.discnp as returned by hmm().
warn	Logical scalar. See the help for sp().
drop	Logical scalar. If there is a single sequence of observations (i.e. if object[["y"]] consists of a matrix or a list of length 1) and if drop is TRUE then the returned value is a single entity (matrix, list of two matrices, or 3-dimensional array, depending on circumstances. Otherwise the returned value is a list of such entities, one for each observation sequence.
...	Not used.

Details

The observation sequence(s) must be present in object (which will be the case if object was returned by hmm() and if the argument keep.y was set to TRUE). If it is not present an error is thrown.

However, if such an error is thrown, do not despair! You *do not* have to start from scratch when fitting your model with keep.y==TRUE. If fit is your fitted model that you obtained *without* setting keep.y==TRUE, then you can just re-fit the model using fit as the starting values:

```
fit2 <- hmm(<whatever>,par0=fit,keep.y=TRUE)
```

This will of course converge instantaneously. You could also do:

```
fit2 <- update(fit,data=<whatever>,keep.y=TRUE)
```

Value

If the observations consist of a single sequence and if drop is TRUE then the returned value consists of a single object (matrix, list of two matrices, or 3-dimensional array, depending on circumstances; see below). Otherwise the returned value is a list of such objects, one for each observation sequence.

If the observations are numeric then the object corresponding to each observation sequence is a matrix. If the model is univariate (see hmm()) then matrix has a single column constituting the sequence of fitted values corresponding to the observations in the given sequence. The number of rows is the number of observations and the entry in row t is the fitted value (conditional expection) corresponding to the observation made at time t. If the model is bivariate (either independent or dependent) then the matrix has two columns corresponding respectively to the two variables in the bivariate model.

If the observations are categorical then the nature of the object returned changes substantially depending on whether the data are univariate, bivariate independent or bivariate dependent. (See hmm().

In the unvariate case the object corresponding to each sequence is a matrix, the number of rows of which is the number of observations and the number of columns of which is the number of unique *possible* values of the observations. The entry of this matrix in row `t` and column `j` is the conditional probability that an emission, at time `t`, is equal to u_i where $u_1, \ldots, u_m$ are the unique possible values.

In the bivariate independent case the object is a *list* of two matrices, each of which is of the same nature as that produced in the univariate case, corresponding respectively to the first and second of the two variables.

In the bivariate dependent case the object is a 3-dimensional array of dimension $m_1 \times m_2 \times n$ where m_1 is the number of unique possible values of the first variable, m_2 is the number of unique possible values of the second variable, and n is the number of observations. The `(i,j,t)-th` entry of this array is the conditional probability that an emission, at time `t`, is equal to (u_i, v_j) where the u_i are the unique possible values of the first variable and the v_j are the unique possible values of the second variable.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

`sp()` `link{predict.hmm.discnp}()`

Examples

```
P <- matrix(c(0.7,0.3,0.1,0.9),2,2,byrow=TRUE)
R <- matrix(c(0.5,0,0.1,0.1,0.3,
              0.1,0.1,0,0.3,0.5),5,2)
set.seed(42)
lll <- sample(250:350,20,TRUE)
y   <- rhmm(ylengths=lll,nsim=1,drop=TRUE,tpm=P,Rho=R)
fit <- hmm(y,K=2,verb=TRUE,keep.y=TRUE,itmax=10)
fv  <- fitted(fit)
```

`hmm` *Fit a hidden Markov model to discrete data.*

Description

Effects a maximum likelihood fit of a hidden Markov model to discrete data where the observations come from one of a number of finite discrete distributions, depending on the (hidden) state of the Markov chain. These distributions (the "emission probabilities") are specified non-parametrically. The observations may be univariate, independent bivariate, or dependent bivariate. By default this function uses the EM algorithm. In the univariate setting it may alternatively use a "brute force" method.

Usage

```
hmm(y, yval=NULL, par0=NULL, K=NULL, rand.start=NULL,
    method=c("EM","bf","LM","SD"), hglmethod=c("fortran","oraw","raw"),
    optimiser=c("nlm","optim"), optimMethod=NULL, stationary=cis,
    mixture=FALSE, cis=TRUE, indep=NULL, tolerance=1e-4, digits=NULL,
    verbose=FALSE, itmax=200, crit=c("PCLL","L2","Linf","ABSGRD"),
    X=NULL,keep.y=FALSE, keep.X=keep.y,
    addIntercept=TRUE, lmc=10, hessian=FALSE,...)
```

Arguments

y	A vector or a list of vectors, or one or two column matrix (bivariate setting) or a list of such matrices; missing values are allowed. If y is a vector, or list of vectors (of discrete data) these vectors are coerced to one column matrices. The entries of these vectors or matrices may be numeric or character and are assumed to constitute discrete data.
yval	A vector (of length m, say) of possible values for the data or a list of two such vectors (of lengths m1 and m2, say, one for each of the two variates in the bivariate settings). These vectors default to the sorted unique values of the respective variates as provided in y. If yval is supplied and any value of y does not match some value of yval, then an error is thrown. The argument yval is provided so as to allow for fitting of models to data in which some of the data values "of interest" were never observed. The estimated emission probabilities of such "never observed" values will of course be zero.
par0	An optional (*named*) list of starting values for the parameters of the model, with components tpm (transition probability matrix), *optionally* ispd (initial state probability distribution) and Rho. The object Rho specifies the probability that the observations take on each of the possible values of the variate or variates, given the state of the hidden Markov chain. See **Details**. Note that in the case of independent bivariate data Rho is a list of two matrices. These matrices may (and in general will) have different row dimensions, but must have identical column dimensions (equal to K, the number of states; see below). If the model is stationary (i.e. if stationary is TRUE) then you should almost surely not specify the ispd component of par0. If you do specify it, it really only makes sense to specify it to be the stationary distribution determined by tpm and this is a waste of time since this is what the code will take ispd to be if you leave it unspecified. If par0 is not specified, starting values are created by the (undocumented) function init.all().
K	The number of states in the hidden Markov chain; if par0 is not specified K *MUST* be; if par0 is specified, K is ignored. Note that K=1 is acceptable; if K is 1 then all observations are treated as being independent and the non-parametric estimate of the distribution of the observations is calculated in the "obvious" way.
rand.start	Either a logical scalar or a list consisting of two logical scalars which must be named tpm and Rho. If the former, it is converted internally into a list with

entries named tpm and Rho, both having the same value as the original argument. If tpm is TRUE then the function init.all() chooses entries for the starting value of tpm at random; likewise for Rho. If left NULL, this argument defaults to list(tpm=FALSE,Rho=FALSE).

method — Character string, either "bf", "EM", "LM" or "SD" (i.e. use numerical maximisation via either nlm() or optim(), the EM algorithm, the Levenberg-Marquardt algorithm, or the method of steepest descent). May be abbreviated. Currently the "bf", "LM" and "SD" methods can be used only in the univariate setting, handle only stationary models (see below) and do not do mixtures.

hglmethod — Character string; one of "fortran", "oraw" or "raw". May be abbreviated. This argument determines the procedure by which the hessian, gradient and log likelihood of the model and data are calculated. If this is argument is equal to "fortran" (the default) then (obviously!) dynamically loaded fortran subroutines are used. The other two possibilities effect the calculations in raw R; "oraw" ("o" for "original" uses code that is essentially a direct transcription of the fortran code, do-loops being replaced by for-loops. With method "raw" the for-loops are eliminated and matrix-vector calculations are applied. The "oraw" method is about 25 times slower than the "fortran" method and the "raw" method is (surprisingly?) even worse; it is more than 30 times slower. The "raw" methods are present mainly for debugging purposes and would not usually be used in practice. This argument is used only if the method is "LM" or "SD" (and is involved only peripherally in the latter instance). It is ignored otherwise.

optimiser — Character string specifying the optimiser to use when the ""bf"" method of optimisation is chosen. It should be one of "nlm" or "optim", and may be abbreviated. Ignored unless method="bf".

optimMethod — Character string specifying the optimisation method to be used by optim(). Should be one of "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", or "Brent". Ignored if the method is not "bf" or if the optimiser is not "optim".

stationary — Logical scalar. If TRUE then the model is fitted under the stationarity assumption, i.e. that the Markov chain was in steady state at the time that observations commenced. In this case the initial state probability distribution is estimated as the stationary distribution determined by the (estimated) transition probability matrix. Otherwise if cis (see below) is TRUE the initial state probability distribution is estimated as the mean of the vectors of conditional probabilities of the states, given the observation sequences, at time t=1. If stationary is TRUE and cis is FALSE an error is thrown. Currently if the method is "bf", "LM" or "SD", and stationary is FALSE, then an error is thrown.

mixture — A logical scalar; if TRUE then a mixture model (all rows of the transition probability matrix are identical) is fitted rather than a general hidden Markov model. Currently an error is thrown if mixture=TRUE and the method is "bf", "LM" or "SD".

cis — A logical scalar specifying whether there should be a **constant initial state** probability distribution. If stationary is FALSE and cis is FALSE then the initial state probability distribution for a given observation sequence is equal to 1 where the (first) maximum of the vector of conditional probabilities of the

states, given the observation sequences, at time t=1, occurs, and is 0 elsewhere. If stationary is TRUE and cis is FALSE an error is given.

indep — Logical scalar. Should the bivariate model be fitted under the assumption that the two variables are (conditionally) independent give the state? If this argument is left as NULL its value is inferred from the structure of Rho in par0 if the latter is supplied. If the data are bivariate and neither indep nor par0 is supplied, then an error is given. If the data are bivariate and if the value of indep is inconsistent with the structure of par0$Rho then an error is given. If the data are univariate then indep is ignored.

tolerance — If the value of the quantity used for the stopping criterion is less than tolerance then the algorithm is considered to have converged. Ignored if method="bf". Defaults to 1e-4.

digits — Integer scalar. The number of digits to which to print out "progress reports" (when verbose is TRUE). There is a "sensible" default (calculated from tolerance). Not used if the method is "bf".

verbose — A logical scalar determining whether to print out details of the progress of the algorithm. If the method is "EM", "LM" or "SD" then when verbose is TRUE information about the convergence criteria is printed out at every step that the algorithm takes. If method="bf" then the value of verbose determines the value of the argument print.level of nlm() or the value of the argument trace of optim(). In the first case, if verbose is TRUE then print.level is set to 2, otherwise it is set to 0. In the second case, if verbose is TRUE then trace is set to 6, otherwise it is set to 0.

itmax — When the method is "EM", "LM" or "SD" this is the maximum number of steps that the algorithm takes. If the convergence criterion has not been met by the time itmax steps have been performed, a warning message is printed out, and the function stops. A value is returned by the function anyway, with the logical component converged set to FALSE. When method="bf" the itmax argument is passed to nlm() as the value of iterlim or to optim() as the value of maxit. If the (somewhat obscure) convergence criteria of nlm() or optim() have not been met by the time itmax "iterations" have been performed, the algorithm ceases. In this case, if nlm() is used. the value of code in the object returned set equal to 4 and if optim() is used then the value of convergence returned is set equal to 1. Note that the value of code, respectively convergence is returned as the converged component of the object returned by hmm(). A value of 1 indicates successful completion of the nlm() procedure. A value of 0 indicates successful completion of the optim() procedure.

crit — The name of the stopping criterion used. When method="EM" it must be one of "PCLL" (percent change in log-likelihood; the default), "L2" (L-2 norm, i.e. square root of sum of squares of change in coefficients), or "Linf" (L-infinity norm, i.e. maximum absolute value of change in coefficients). When method="LM" or method="SD" there is a fourth possibility, namely "ABSGRD" the (maximum) absolute value of the gradient. It may not be advisable to use this criterion in the current context (i.e. that of discrete non-parametric distributions). See **Warnings**. This argument defaults to "PCLL". It is ignored if method="bf". (The nlm() and optim() functions have their own obscure stopping criteria.)

`X` An optional *numeric* matrix, or a list of such matrices, of "auxiliary" *predictors*. The use of such predictors is (currently, at least) applicable only in the univariate emissions setting. If `X` is a list it must be of the same length as `y` and all entries of this list must have the same number of columns. If the columns of any entry of the list are named, then they must be named for *all* entries, and the column names must be the *same* for all entries. The number of rows of each entry must be equal to the length of the corresponding entry of `y`. If `X` is a matrix then `y` should be a vector or one-column matrix (or a list with a single entry equal to such).

There may be at most one constant column in `X` or the components thereof. If there are *any* constant columns there must be precisely one (in all components of `X`), it must be the first column and all of its entries must be equal to `1`. If the columns have names, the names of this first column must be `"Intercept"`.

Note that `X` (or its entries) must be a *numeric* matrix (or must be numeric matrices) — not data frames! Factor predictors are not permitted. It may be possible to use factor predictors by supplying `X` or its entries as the output of `model.matrix()`; this will depend on circumstances.

The fitted coefficients that are produced when `X` is supplied, are (to put it mildly) a bit difficult to interpret. See **Fitted Coefficients of Auxiliary Predictors** for some discussion.

`keep.y` Logical scalar; should the observations `y` be returned as a component of the value of this function?

`keep.X` Logical scalar; should the predictors `X` be returned as a component of the value of this function? Note that the value of `keep.X` will be silently set equal to `FALSE` unless it actually "makes sense" to keep `X`. I.e. unless the observations are *univariate* and `X` is actually supplied, i.e. is not `NULL`.

`addIntercept` Logical scalar. Should a column of ones, corresponding to an intercept term, be prepended to each of the matrices in the list `X`? If each of these matrices already has an initial column of ones, then setting `addIntercept=TRUE` results in an error being thrown. If this is not the case, then by default an initial column of ones is added.

`lmc` Numeric scalar. The (initial) "Levenberg-Marquardt correction" parameter. Used only if `method="LM"`, otherwise ignored.

`hessian` Logical scalar. Should the hessian matrix be returned? This argument is relevant only if `method="bf"` (in which case it is passed along to `hmmNumOpt()`) and is ignored otherwise. This argument should be set to `TRUE` only if you *really* want the hessian matrix. Setting it to `TRUE` causes a substantial delay between the time when `hmm()` finishes its iterations and when it actually returns a value.

`...` Additional arguments passed to `hmmNumOpt()`. There is one noteworthy argument `useAnalGrad` which is used "directly" by `hmmNumOpt()`. This argument is a logical scalar and if it is `TRUE` then calls to `nlm()` or `optim()` are structured so that an analytic calculation of the gradient vector (implemented by the internal function `get.gl()` is applied. If it is `FALSE` then finite difference methods are used to calculate the gradient vector. If this argument is not specified it defaults to `FALSE`. Note that the name of this argument **cannot be abbreviated**.

Other "additional arguments" may be supplied for the control of `nlm()` and are passed on appropriately to `nlm()`. These are used only if `method="bf"` and if

optimiser="nlm". These "..." arguments might typically include gradtol, stepmax and steptol. They should **NOT** include print.level or iterlim. The former argument is automatically passed to nlm() as 0 if verbose is FALSE and as 2 if verbose is TRUE. The latter argument is automatically passed to nlm() with the value of itmax.

Details

- **Univariate case:** In the univariate case the emission probabilities are specified by means of a data frame Rho. The first column of Rho, named "y", is a factor consisting of the possible values of the emissions, repeated K times (where K is the number of states). The second column, named states, is a factor consisting of integer values 1, 2, ..., K. Each of these values is repeated m times where m is the length of yval. Further columns of Rho are numeric and consist of coefficients of the linear predictor of the probabilities of the various values of y. If X is NULL then Rho has only one further column named Intercept.

  If X is not NULL then the Intercept column is present only if addIntercept is TRUE. There as many (other, in addition to the possible Intercept column) numeric columns as there are columns in X or in the matrices in the list X. The names of these columns are taken to be the column names of X or the *first* entry of X if such column names are present. Otherwise the names default to V1, V2 ....

  The probabilities of the emissions taking on their various possible values are given by

$$\Pr(Y = y_i|\boldsymbol{x}, \text{state} = S) = \ell_i / \sum_{j=1}^{m} \ell_j$$

  where ℓ_j is the jth entry of $\boldsymbol{\beta}^\top \boldsymbol{x}$ and where in turn $\boldsymbol{x}$ is the vector of predictors and $\boldsymbol{\beta}$ is the coefficient vector in the linear predicator that corresponds to y_i and the hidden state S. For identifiability the vectors $\boldsymbol{\beta}$ corresponding to the first value of Y (the first level of Rho$y) are set equal to the zero vector for all values of the state S.

  Note that the Rho component of the starting values par0 may be specified as a *matrix* of probabilities, with rows corresponding to possible values of the observations and columns corresponding to states. That is the Rho component of par0 may be provided in the form Rho $= [\rho_{ij}]$ where $\rho_{ij} = \Pr(Y = y_i|S = j)$. This is permissible as long as X is NULL and may be found to be more convenient and intuitive. If the starting value for Rho is provided in matrix form it is (silently) converted internally into the data frame form, by the (undocumented) function cnvrtRho().

  When argument X is *not* NULL, it is difficult to specify a "reasonable" value for the Rho component of par0. One might try to specify par0$Rho in the data frame form. The question of how to specify the columns of par0$Rho corresponding to the auxiliary predictors (columns of X or of the entries of X) is a thorny one.

  It is permissible in these circumstances to specify par0$Rho as a matrix of probabilities, just as one would do if X were NULL. In this setting the (undocumented) function checkStartVal() converts the matrix of probabilities to data frame form and then appends columns, all of whose entries are 0, corresponding to the auxiliary predictors. When par0 is unspecified, the (undocumented) function init.all() performs similar construction to accommodate a non-NULL value of X. Whether the resulting starting value for Rho makes any real sense, is questionable. However little else can be done.

- **Independent bivariate case:** the emission probabilities are specified by a list of two matrices. In this setting $\Pr(Y_1, Y_2) = (y_{i1}, y_{i2})|S = j) = \rho^{(1)}_{i_1,j}\rho^{(2)}_{i_2,j}$ where $R^{(k)} = [\rho^{(k)}_{ij}]$ $(k = 1, 2)$ are the two emission probability matrices.
- **Dependent bivariate case:** the emission probabilities are specified by a three dimensional array. In this setting $\Pr((Y_1, Y_2) = (y_{i1}, y_{i2})|S = j) = \rho_{i_1,i_2,j}$ where $R = [\rho_{ijk}]$ is the emission probability array.

The hard work of calculating the recursive probabilities used to fit the model is done by a Fortran subroutine `recurse` (actually coded in Ratfor) which is dynamically loaded. In the univariate case, when `X` is provided, the estimation of the "linear predictor" vectors $\boldsymbol{\beta}$ is handled by the function `multinom()` from the `nnet` package. Note that this is a "Recommended" package and is thereby automatically available (i.e. does not have to be installed).

Value

A list with components:

`Rho` — The fitted value of the data frame, list of two matrices, or array `Rho` (in the case of a univariate model, a bivariate independent model or a bivariate dependent model respectively) specifying the distributions of the observations (the "emission" probabilities).

`Rho.matrix` — Present *only* in the univariate setting. A matrix whose entries are the (fitted) emission probabilities, row corresponding to values of the emissions and columns to states. The columns sum to 1. This component provides the same information as `Rho`, but in a more readily interpretable form.

`tpm` — The fitted value of the transition probability matrix `tpm`.

`stationary` — Logical scalar; the value of the `stationary` argument.

`ispd` — The fitted initial state probability distribution, or a matrix of initial state probability distributions, one (column) of `ispd` for each observation sequence.

If `stationary` is `TRUE` then `ispd` is assumed to be the (unique) stationary distribution for the chain, and thereby determined by the transition probability matrix `tpm`. If `stationary` is `FALSE` and `cis` is `TRUE` then `ispd` is estimated as the mean of the vectors of conditional probabilities of the states, given the observation sequences, at time `t=1`.

If `cis` is `FALSE` then `ispd` is a matrix whose columns are the vectors of conditional probabilities of the states, given the observation sequences, at time `t=1`, as described above. (If there is only one observation sequence, then this — one-column — matrix is converted into a vector.)

`log.like` — The final (maximal, we hope!) value of the log likelihood, as determined by the maximisation procedure.

`grad` — The gradient of the log likelihood. Present only if the method is `"LM"` or `"bf"` and in the latter case then only if the optimiser is `nlm()`.

`hessian` — The hessian of the log likelihood. Present only if the method is `"LM"` or `"bf"`.

`stopCrit` — A vector of the (final) values of the stopping criteria, with names `"PCLL"`, `"L2"`, `"Linf"` unless the method is `"LM"` or `"SD"` in which case this vector has a fourth entry named `"ABSGRD"`.

`par0` The starting values used by the algorithms. Either the argument `par0`, or a similar object with either or both components (`tpm` and `Rho`) being created by `rand.start()`.

`npar` The number of parameters in the fitted model. Equal to `nispar + ntpmpar + nrhopar` where (1) `nispar` is `0` if `stationary` is `TRUE` and is `K-1` otherwise; (2) `ntpmpar` is `K*(K-1)` (3) `nrhopar` is

- `(nrow(Rho) - K)*(ncol(Rho)-2)` for univariate models
- `K*(sum(sapply(Rho,nrow))-K)` for bivariate independent models
- `prod(dim(Rho))-K` for bivariate dependent models.

`bicm` Numeric scalar. The number by which `npar` is multiplied to form the `BIC` criterion. It is essentially the log of the number of observations. See the code of `hmm()` for details.

`converged` A logical scalar indicating whether the algorithm converged. If the EM, LM or steepest descent algorithm was used it simply indicates whether the stopping criterion was met before the maximum number (`itmax`) of steps was exceeded. If `method="bf"` then `converged` is based on the `code` component of the object returned by the optimiser when `nlm()` was used, or on the `convergence` component when `optim()` was used. In these cases `converged` has an *attribute* (`code` or `convergence` respectively) giving the (integer) value of the relevant component.

Note that in the `nlm()` case a value of `code` equal to 2 indicates "probable" convergence, and a value of 3 indicates "possible" convergence. However in this context `converged` is set equal to `TRUE` *only* if `code` is 1.

`nstep` The number of steps performed by the algorithm if the method was `"EM"`, `"LM"` or `"SD"`. The value of `nstep` is set equal to the `iterations` component of the value returned by `nlm()` if `method="bf"`.

`prior.emsteps` The number of EM steps that were taken before the method was switched from `"EM"` to `"bf"` or to `"LM"`. Present only in values returned under the `"bf"` or `"LM"` methods after a switch from `"EM"` and is equal to `0` if either of these methods was specified in the initial call (rather than arising as the result of a switch).

`ylengths` Integer vector of the lengths of the observation sequences (number of rows if the observations are in the form of one or two column matrices).

`nafrac` A real number between 0 and 1 or a pair (two dimensional vector) of such numbers. Each number is the the fraction of missing values if the corresponding components of the observations.

`y` An object of class `"tidyList"`. It is a tidied up version of the observations; i.e. the observations `y` after the application of the undocumented function `tidyList()`. Present only if `keep.y` is `TRUE`.

`X` An object of class `"tidyList"`. It is tidied up version of the predictor matrix or list of predictor matrices; i.e. the argument `X` after the application of `tidyList()` (with argument `rp` set to `"predictor"`. Present only if `X` is supplied, is an appropriate argument, and if `keep.X` is `TRUE`.

`parity` Character string; `"univar"` if the data were univariate, `"bivar"` if they were bivariate.

`numeric` Logical scalar; `TRUE` if the (original) data were numeric, `FALSE` otherwise.

AIC — The value of AIC = `-2*log.like + 2*npar` for the fitted model.

BIC — The value of BIC = `-2*log.like + log(nobs)*npar` for the fitted model. In the forgoing `nobs` is the number of observations. This is the number of *non-missing* values in `unlist(y)` in the univariate setting and one half of this number in the bivariate setting.

args — A list of argument values supplied. This component is returned in the interest of making results reproducible. It is also needed to facilitate the updating of a model via the update method for the class `hmm.discnp`, `update.hmm.discnp()`.

It has components:

- `method`
- `optimiser`
- `optimMethod`
- `stationary`
- `mixture`
- `cis`
- `tolerance`
- `itmax`
- `crit`
- `addIntercept`

Thanks

A massive nest of bugs was eliminated in the transition from version 3.0-8 to version 3.0-9. These bugs arose in the context of using auxiliary predictor variables (argument `X`). The handling of such auxiliary predictors was completely messed up. I am grateful to Leah Walker for pointing out the problem to me.

Warnings

The ordering of the (hidden) states can be arbitrary. What the estimation procedure decides to call "state 1" may not be what *you* think of as being state number 1. The ordering of the states will be affected by the starting values used.

Some experiences with using the `"ABSGRD"` stopping criterion indicate that it may be problematic in the context of discrete non-parametric distributions. For example a value of 1854.955 was returned after 200 LM steps in one (non-convergent, of course!) attempt at fitting a model. The stopping criterion `"PCLL"` in this example took the "reasonable" value of 0.03193748 when iterations ceased.

Notes — Various

This function *used* to have an argument `newstyle`, a logical scalar (defaulting to `TRUE`) indicating whether (in the univariate setting) the emission probabilities should be represented in "logistic" form. (See **Details**, **Univariate case:**, above.) Now the emission probabilities are *always* represented in the "logistic" form. The component `Rho` of the starting parameter values `par0` may still be supplied as a matrix of probabilities (with columns summing to 1), but this component is converted (internally, silently) to the logistic form.

The object returned by this function also has (in the univariate setting), in addition to the component Rho, a component Rho.matrix giving the emission probabilities in the more readily interpretable matrix-of-probabilities form. (See **Value** above.)

The package *used* to require the argument y to be a *matrix* in the case of multiple observed sequences. If the series were of unequal length the user was expected to pad them out with NAs to equalize the lengths.

The old matrix format for multiple observation sequences was permitted for a while (and the matrix was internally changed into a list) but this is no longer allowed. If y is indeed given as a matrix then this corresponds to a single observation sequence and it must have one (univariate setting) or two (bivariate setting) columns which constitute the observations of the respective variates.

If K=1 then tpm, ispd, converged, and nstep are all set equal to NA in the list returned by this function.

The estimate of ispd in the non-stationary setting is inevitably very poor, unless the number of sequences of observations (the length of the list y) is very large. We have in effect "less than one" relevant observation for each such sequence.

The returned values of tpm and Rho (or the entries of Rho when Rho is a list) have dimension names. These are formed from the argument yval if this is supplied, otherwise from the sorted unique values of the observations in y. Likewise the returned value of ispd is a named vector, the names being the same as the row (and column) names of tpm.

If method is equal to "EM" there may be a *decrease* (!!!) in the log likelihood at some EM step. This is "theoretically impossible" but can occur in practice due to an intricacy in the way that the EM algorithm treats ispd when stationary is TRUE. It turns out to be effectively impossible to maximise the expected log likelihood unless the term in that quantity corresponding to ispd is ignored (whence it *is* ignored). Ignoring this term is "asymptotically negligible" but can have the unfortunate effect of occasionally leading to a decrease in the log likelihood.

If such a decrease is detected, then the algorithm terminates and issues a message to the effect that the decrease occurred. The message suggests that another method be used and that perhaps the results from the penultimate EM step (which are returned by this function) be used as starting values.

It seems to me that it *should* be the case that such a decrease in the log likelihood can occur only if stationary is TRUE. However I have encountered instances in which a decrease occurred when stationary was FALSE. I have yet to figure out/track down what is going on here.

Note on method

If the method is "EM" it is actually possible for the log likelihood to *decrease* at some EM step. This is "impossible in an ideal world" but can happen to the fact the EM algorithm, as implemented in this package at least, cannot maximise the expected log likelihood if the component corresponding to the initial state probability distribution is taken into consideration. This component should ideally be maximised subject to the constraint that t(P)%*%ispd = ispd, but this constraint seems to effectively impossible to impose. Lagrangian multipliers don't cut it. Hence the summand in question is ignored at the M-step. This usually works alright since the summand is asymptotically negligible, but things can sometimes go wrong. If such a decrease occurs, an error is thrown.

In previous versions of this package, instead of throwing an error the hmm() function would automatically switch to either the "bf" or the "LM" method, depending whether a matrix X of auxiliary

predictors is supplied, starting from the penultimate parameter estimates produced by the EM algorithm. However this appears not to be a good idea; those "penultimate estimates" appear not to be good starting values for the other methods. Hence an error is now thrown and the user is explicitly instructed to invoke a different method, "starting from scratch".

Fitted Coefficients of the Predictors

It is of course of interest to understand the meaning of the coefficients that are fitted to the predictors in the model. If `X` is supplied then the number of predictors is (as a rule) one (for the intercept) plus the number of columns in each entry of `X`. We say "as a rule" because, e.g., the entries of `X` could each have an "intercept" column, or the `addIntercept` argument could be `FALSE`. If `X` is not supplied there is only one predictor, named `Intercept`.

The interpretation of these predictor coefficients is a bit subtle. To get an idea of what it's all about, consider the output from example `4`. (See **Examples**). The fitted coefficients in question are to be found in columns 3 and onward of the component `Rho` of the object returned by `hmm()`. In the context of example `4`, this object is `fit.wap`. (The suffix `wap` stands for "with auxiliary predictors".)

```
fit.wap$Rho
     y state Intercept     ma.com      nh.com     bo.com
1   lo     1 1.3810463  0.4527982 -3.27161353 -1.9563915
2  mlo     1 0.1255631 -1.1402546 -1.37713744  0.5946980
3    m     1 0.7356526  0.1523734 -2.70841817 -0.1794645
4  mhi     1 0.8479798 -0.2438988 -1.12544989 -0.9650320
5   hi     1 0.0000000  0.0000000  0.00000000  0.0000000
6   lo     2 3.9439410 -0.8355306 -0.77702276  1.4963631
7  mlo     2 2.6189880 -1.9373885 -0.09190623  0.8316870
8    m     2 2.1457317 -1.7276183  0.19524655 -0.3249485
9  mhi     2 1.8834139 -1.3760011 -0.59806309  1.2828365
10  hi     2 0.0000000  0.0000000  0.00000000  0.0000000
```

If you multiply the matrix consisting of the predictor coefficients (columns 3 to 6 of `Rho` in this instance) times a vector of predictors you get, for each state, the "exponential form" of the probabilities ("pre-probabilities") for each of the possible `y`-values, given the vector of predictors.

E.g. set `x <- c(1,1,0,0)`. This vector picks up the intercept and indicates that the Malabar outfall has been commissioned, the North Head outfall has not been commissioned, and the Bondi Offshore outfall has not been commissioned.

Now set:

```
pp1 <- (as.matrix(fit.wap$Rho)[,3:6]%*%x)[1:5]
pp2 <- (as.matrix(fit.wap$Rho)[,3:6]%*%x)[6:10]
```

Note that `pp1` consists of "exponential probabilities" corresponding to state 1, and `pp2` consists of "exponential probabilities" corresponding to state 2. To convert the foregoing pre-probabilities to the actual probabilities of the `y`-values, we apply the — undocumented — function `expForm2p()`:

```
p1 <- expForm2p(pp1)
p2 <- expForm2p(pp2)
```

The value of `p1` is

```
[1] 0.52674539 0.03051387 0.20456767 0.15400019 0.08417288
```

and that of p2 is

```
[1] 0.78428283 0.06926632 0.05322204 0.05819340 0.03503541
```

Note that p1 and p2 each sum to 1, as they should/must do. This says, e.g., that when the system is in state 2, and Malabar has been commissioned but North Head and Bondi Offshore have not, the (estimated) probability that y is "mhi" (medium-high) is 0.05819340.

It may be of some interest to test the hypothesis that the predictors have any actual predictive power at all:

```
fit.nap <- hmm(xxx,yval=Yval,K=2,verb=TRUE)
# "nap" <--> no aux. preds
```

There is a bit of a problem here, in that the likelihood *decreases* at EM step 65. (See the warning message.)

We can check on this problem by refitting using method="LM".

```
fit.nap.lm <- hmm(xxx,yval=Yval,par0=fit.nap,method="LM",verb=TRUE)
```

Doing so produces only a small improvement in the log likelihood (from -1821.425 to -1820.314), so we really could have ignored the problem. We can now do anova(fit.wap,fit.nap) which gives

```
$stat
[1] 153.5491

$df
[1] 24

$pvalue
[1] 7.237102e-21
```

Thus the p-value is effectively zero, saying that in this instance the auxiliary predictors appear to have a "significant" impact on the fit.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

References

Rabiner, L. R., "A tutorial on hidden Markov models and selected applications in speech recognition," Proc. IEEE vol. 77, pp. 257 – 286, 1989.

Zucchini, W. and Guttorp, P., "A hidden Markov model for space-time precipitation," Water Resources Research vol. 27, pp. 1917-1923, 1991.

MacDonald, I. L., and Zucchini, W., "Hidden Markov and Other Models for Discrete-valued Time Series", Chapman & Hall, London, 1997.

Liu, Limin, "Hidden Markov Models for Precipitation in a Region of Atlantic Canada", Master's Report, University of New Brunswick, 1997.

See Also

rhmm(), mps(), viterbi()

Examples

```
# TO DO: Create one or more bivariate examples.
#
# The value of itmax in the following examples is so much
# too small as to be risible.  This is just to speed up the
# R CMD check process.
# 1.
Yval <- LETTERS[1:10]
Tpm  <- matrix(c(0.75,0.25,0.25,0.75),ncol=2,byrow=TRUE)
Rho  <- cbind(c(rep(1,5),rep(0,5)),c(rep(0,5),rep(1,5)))/5
rownames(Rho) <- Yval
set.seed(42)
xxx  <- rhmm(ylengths=rep(1000,5),nsim=1,tpm=Tpm,Rho=Rho,yval=Yval,drop=TRUE)
fit  <- hmm(xxx,par0=list(tpm=Tpm,Rho=Rho),itmax=10)
print(fit$Rho) # A data frame
print(cnvrtRho(fit$Rho)) # A matrix of probabilities
                         # whose columns sum to 1.

# 2.
# See the help for logLikHmm() for how to generate y.num.
## Not run:
   fit.num     <- hmm(y.num,K=2,verb=TRUE,itmax=10)
   fit.num.mix <- hmm(y.num,K=2,verb=TRUE,mixture=TRUE,itmax=10)
   print(fit.num[c("tpm","Rho")])

## End(Not run)
# Note that states 1 and 2 get swapped.

# 3.
xxx <- with(SydColDisc,split(y,f=list(locn,depth)))
Yval <- c("lo","mlo","m","mhi","hi")
# Two states: above and below the thermocline.
fitSydCol <- hmm(xxx,yval=Yval,K=2,verb=TRUE,itmax=10)

# 4.
X <- split(SydColDisc[,c("ma.com","nh.com","bo.com")],
           f=with(SydColDisc,list(locn,depth)))
X <- lapply(X,function(x){
                as.matrix(as.data.frame(lapply(x,as.numeric)))-1})
fit.wap <- hmm(xxx,yval=Yval,K=2,X=X,verb=TRUE,itmax=10)
# wap <--> with auxiliary predictors.
```

```
# 5.
## Not run:  # Takes too long.
fitlm <- hmm(xxx,yval=Yval,K=2,method="LM",verb=TRUE)
fitem <- hmm(xxx,yval=Yval,K=2,verb=TRUE)
# Algorithm terminates due to a decrease in the log likelihood
# at EM step 64.
newfitlm <- hmm(xxx,yval=Yval,par0=fitem,method="LM",verb=TRUE)
# The log likelihood improves from -1900.988 to -1820.314

## End(Not run)

# 6.
fitLesCount <- hmm(lesionCount,K=2,itmax=10) # Two states: relapse and remission.
```

`hydroDat` *Canadian hydrological data sets.*

Description

Five data sets obtained from the "HYDAT" database, Environment and Climate Change Canada's database of historical hydrometric data. The data were obtained using the `tidyhydat` package. The data have been trimmed so that there are no gaps in the observation dates and are presented in "raw" form and in discretised form as deciles of the residuals (difference between raw values and the daily mean over years).

Usage

```
data("linLandFlows")
data("ftLiardFlows")
data("portMannFlows")
data("portMannSedLoads")
data("portMannSedCon")
```

Format

Data frames with observations on the following 3 variables.

`Date` Dates on which observations were made.

`Value` Numeric vector of observation values.

`mean` The mean over years of `Value`.

`resid` The difference `Value - mean`.

`deciles` A factor with levels `d1`, ..., `d10`, which are the deciles of the variable `resid`

Details

The variable mean was calculated as follows:

```
yday <- as.POSIXlt(X$Date)$yday
mn   <- tapply(X$Value,yday,mean,na.rm=TRUE)
mean <- mn[as.character(yday)]
```

where X is the data set being processed.

The data set linLandFlows originally consisted of 2008 observations; there were 1980 observations after "trimming". The data set ftLiardFlows originally consisted of 22364 observations; there were 11932 observations after "trimming". The data set portMannFlows originally consisted of 6455 observations; there were 3653 observations after "trimming". The data set portMannSedLoads consists of 2771 observations; no observations were trimmed. The data set portMannSedCon consists of 4597 observations; no observations were trimmed.

The units of the "Flows" variables are cubic metres per second (m^3/s); the units of "portMannSedLoads" are tonnes; the units of "portMannSedCon" are milligrams per litre (mg/l).

The "linLandFlows" data were obtained at the Lindberg Landing hydrometric station on the Liard River in the Northwest Territories of Canada. The "ftLiardFlows" data were obtained at the Fort Liard hydrometric station on the Liard River in the Northwest Territories of Canada. The "portMann" data were obtained at the hydrometric station located at the Port Mann pumping station on the Fraser River in the Province of British Columbia in Canada.

Source

Environment and Climate Change Canada's database "HYDAT", a database of historical hydrometric data. The data were obtained vis the tidyhydat package, which is available from "CRAN", https://cran.r-project.org

Examples

```
fit <- hmm(linLandFlows$deciles,K=4,itmax=10)
```

---

lesionCount *Multiple sclerosis lesion counts for three patients.*

Description

Lesion counts for three multiple sclerosis patients. The counts were obtained by magnetic resonance imaging, and were observed at monthly intervals.

Usage

```
lesionCount
```

Format

A list with three components each component being the sequence of counts for a given patient and consisting of a vector with non-negative integer entries.

Modelling

The hidden Markov models applied to these data by Albert et al. and by MacKay and Petkau were much more complex and elaborate than those fitted in the examples in this package. See the references for details.

Source

The data were originally studied by Albert et al., (1994). They are were also analyzed by Altman and Petkau (2005). The data were kindly provided by Prof. Altman.

References

Albert, P. S., McFarland, H. F., Smith, M. E., and Frank, J. A. Time series for modelling counts from a relapsing-remitting disease: application to modelling disease activity in multiple sclerosis. *Statistics in Medicine* **13** (1994) 453–466.

Altman, Rachel MacKay, and Petkau, A. John. Application of hidden Markov models to multiple sclerosis lesion count data. *Statistics in Medicine* **24** (2005) 2335–2344.

---

logLikHmm *Log likelihood of a hidden Markov model*

Description

Calculate the log likelihood of a hidden Markov model with discrete non-parametric observation distributions.

Usage

```
logLikHmm(y, model=NULL, tpm=NULL, ispd=NULL, Rho=NULL, X=NULL,
          addIntercept=NULL, warn=TRUE)
```

Arguments

y	A vector, or list of vectors, or a one or two column matrix or a list of such matrics, whose entries consist of observations from a hidden Markov model with discrete non-parametric observation distributions.
model	An object specifying a hidden Markov model, usually returned by hmm().
tpm	The transition probability matrix of the Markov chain. Ignored (and extracted from model) if model is non-NULL.

ispd	The vector of probabilities specifying the initial state probability distribution, or a matrix each of whose columns is a trivial ("delta function") vector specifying the "most probable" initial state for each observation sequence. If ispd is missing then ispd is calculated as the stationary distribution determined by tpm. Ignored (and extracted from model) if model is non-NULL.
Rho	An object specifying the "emission" probabilities of the observations. (See the **Details** in the help for hmm().) Ignored (and extracted from model) if model is non-NULL.
X	An optional *numeric* matrix, or a list of such matrices, of *predictors*. The use of such predictors is (currently, at least) applicable only in the univariate emissions setting. If X is a list it must be of the same length as y and all entries of this list must have the same number of columns. The number of rows of each entry must be equal to the length of the corresponding entry of y. If X is a matrix then y should be a vector or one-column matrix (or a list with a single entry equal to such).
addIntercept	Logical scalar. See the documentation of hmm(). If this argument is not specified, and if model is NULL then an error is thrown.
warn	Logical scalar; should a warning be issued if Rho hasn't got relevant dimension names? (Note that if this is so, then the corresponding dimension names are formed from the sorted unique values of y or of the appropriate column(s) of y. And if *this* is so, then the user should be sure that the ordering of the entries of Rho corresponds properly to the the sorted unique values of y.) This argument is passed to the utility function check.yval() which actually issues the warning if warn=TRUE.

Details

If y is not provided the function simply returns the log.like component of model (which could be NULL if model was not produced by hmm().

The observation values (the entries of the vector or matrix y, or of the vectors or matrices which constitute the entries of y if y is a list) must be consistent with the appropriate dimension names of Rho or of its entries when Rho is a list. More specifically, if Rho has dimension names (or its entries have dimension names) then the observation values must all be found as entries of the appropriate dimension name vector. If a vector of dimension names is NULL then the corresponding dimension must be equal to the number of unique observations of the appropriate variate. integers between 1 and nrow(Rho).

Value

The loglikehood of y given the parameter values specified in par.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

References

See hmm() for references.

See Also

hmm(), pr(), sp()

Examples

```
# TO DO:  One or more bivariate examples.
P <- matrix(c(0.7,0.3,0.1,0.9),2,2,byrow=TRUE)
R <- matrix(c(0.5,0,0.1,0.1,0.3,
              0.1,0.1,0,0.3,0.5),5,2)
set.seed(42)
lll   <- sample(250:350,20,TRUE)
set.seed(909)
y.num <- rhmm(ylengths=lll,nsim=1,tpm=P,Rho=R,drop=TRUE)
set.seed(909)
y.let <- rhmm(ylengths=lll,nsim=1,tpm=P,Rho=R,yval=letters[1:5],drop=TRUE)
row.names(R) <- 1:5
ll1   <- logLikHmm(y.num,tpm=P,Rho=R)
row.names(R) <- letters[1:5]
ll2   <- logLikHmm(y.let,tpm=P,Rho=R)
ll3   <- logLikHmm(y.let,tpm=P,Rho=R,ispd=c(0.5,0.5))
fit   <- hmm(y.num,K=2,itmax=10)
ll4   <- logLikHmm(y.num,fit) # Use the fitted rather than the "true" parameters.
```

---

misstify *Insert missing values.*

Description

Insert missing values into data simulated by rhmm.

Usage

```
misstify(y, nafrac, fep = NULL)
```

Arguments

y	A data set (vector or matrix with one or two columns, whose entries consitute discrete data, or a list of such vectors or matrices) or a list of such data sets (objects of class "multipleHmmDataSets" such as might be generated by rhmm()
nafrac	A numeric vector, some entries of which could be ignored. (See below.) Those which do not get ignored must be probabilities *strictly* less than 1. (Having *everything* missing makes no sense!) The vector nafrac will be replicated to have an "appropriate" length. If y is of class "multipleHmmDataSets" then this length is length(y) if the data are univariate and is 2*length(y) if the data are bivariate. In the former case the entries of the replicated vector from the fraction of missing values in the corresponding data set. In the latter case the odd numbered entries form the fraction of missing values for the first variable and the even numbered entries the fraction

for the second variable. If y is not of class "multipleHmmDataSets" then this length is either 1 (univariate case) or 2 (bivariate case).

Note that replication discards entries that are not needed to make up the required length, and such entries are thereby ignored. E.g. rep(c(0.2,0.7,1.6),length=2) yields [1] 0.2 0.7, i.e. the entry 1.6 is ignored.

The fraction(s) of missing values in a given data set may be determined by nafracCalc().

fep

"First entry present". A list with one or two entries, the first being a logical scalar (which might be named "present". If there is a second entry it should be a scalar probability (which might be named "p2"). In an application of interest, observation sequences always begin at an observed event, i.e. at a time point at which the "emission" has at least one non-missing value. If fep[[1]] is TRUE the NAs will be inserted in such a way that the resulting data have this characteristic. If fep is left NULL then its first (possibly only) entry is set to TRUE.

For *bivariate* data, fep[[2]] specifies the probabilty that *both* values of the initial pair of observations are non-missing. In this case one of the entries of the initial pair is chosen to be "potentially" missing, with probabilities nafrac/sum(nafrac). This entry is left non-missing with probability fep[[2]]. (The other entry is always left non-missing.)

If the data are univariate or if fep[[1]] is FALSE, then fep[[2]] is ignored. If the data are bivariate and fep[[2]] is not specified, it defaults to the (estimated) conditional probability that both entries of the initial pair of observations are present given that at least one is present, under the assumption of independence of these events. I.e. it is set equal to prod(1-nafrac)/(1-prod(1-nafrac)).

Value

An object with a structure similar to that of y, containing the same data as y but with some of these data having been replaced by missing values (NA). In particular, if y is of class "multipleHmmDataSets" then so is the returned value.

Note that rhmm() calls upon misstify() to effect the replacement of a certain fraction of the simulated observations by missing values. If rhmm() is applied to a fitted model, then by default, this "certain fraction" is determined, using nafracCalc(), from the data set to which the model was fitted.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

rhmm() nafracCalc()

Examples

```
P <- matrix(c(0.7,0.3,0.1,0.9),2,2,byrow=TRUE)
```

```
R <- matrix(c(0.5,0,0.1,0.1,0.3,
              0.1,0.1,0,0.3,0.5),5,2)
set.seed(42)
lll   <- sample(250:350,20,TRUE)
y1    <- rhmm(ylengths=lll,nsim=1,tpm=P,Rho=R)
y1m   <- misstify(y1,nafrac=0.5,fep=list(TRUE))
y2    <- rhmm(ylengths=lll,nsim=5,tpm=P,Rho=R)
set.seed(127)
y2m   <- misstify(y2,nafrac=0.5,fep=list(TRUE))
nafracCalc(y2m) # A list all of whose entries are close to 0.5.
set.seed(127)
y2ma  <- lapply(y2,misstify,nafrac=0.5,fep=list(TRUE))
## Not run:
    nafracCalc(y2ma) # Throws an error.

## End(Not run)
sapply(y2ma,nafracCalc) # Effectively the same as nafracCalc(y2m).
```

`mps` *Most probable states.*

Description

Calculates the most probable hidden state underlying each observation.

Usage

```
mps(y, model = NULL, tpm, Rho, ispd=NULL, warn=TRUE)
```

Arguments

`y`	The observations for which the underlying most probable hidden states are required. May be a sequence of observations in the form of a vector or a one or two column matrix, or a list each component of which constitutes a (replicate) sequence of observations. It may also be an object of class `"multipleHmmDataSets"` as returned by `rhmm()` with `nsim>1`. If `y` is missing, it is extracted from `model` (whence it will *not* be of class `"multipleHmmDataSets"`!) provided that `model` and its `y` component are not `NULL`. Otherwise an error is given.
`model`	An object describing a fitted hidden Markov model, as returned by `hmm()`. In order to make any kind of sense, `model` should bear some reasonable relationship to `y`.
`tpm`	The transition probability matrix for a hidden Markov model; ignored if `model` is non-null. Should bear some reasonable relationship to `y`.
`Rho`	An object specifying the probability distributions of the observations ("emission" probabilities) for a hidden Markov model. See `hmm()`. Ignored if `model` is non-null. Should bear some reasonable relationship to `y`.

ispd — A vector specifying the initial state probability distribution for a hidden Markov model, or a matrix each of whose columns are trivial ("delta function") vectors specifying the "most probable" initial state for each observation sequence.

This argument is ignored if `model` is non-null. It should bear some reasonable relationship to `y`. If both `ispd` and `model` are `NULL` then `ispd` is taken to be the stationary distribution of the chain, calculated from `tpm`.

warn — Logical scalar; in the bivariate setting, should a warning be issued if the two matrices constituting `Rho` (bivariate independent case) or the array constituting `Rho` (bivariate dependent case) have not got relevant dimension names? (Note that if this is so, then the corresponding dimension names are formed from the sorted unique values of the appropriate columns of `y`. And if *this* is so, then the user should be sure that the ordering of the entries of `Rho` corresponds properly to the the sorted unique values of `y`.) This argument is passed to the utility function `check.yval()` which actually issues the warning if `warn=TRUE`.

Details

For each t the maximum value of $\gamma_t(i)$, i.e. of the (estimated) probability that the state at time t is equal to i, is calculated, and the value of the state with the corresponding index is returned.

Value

If `y` is a single observation sequence, then the value is a vector of corresponding most probable states.

If `y` is a list of replicate sequences, then the value is a list, the j-th entry of which constitutes the vector of most probable states underlying the j-th replicate sequence.

If `y` is of class `"multipleHmmDataSets"` then the value returned is a list of lists of the sort described above.

Warning

The *sequence of most probable states* as calculated by this function will not in general be the *most probable sequence of states*. It may not even be a *possible* sequence of states. This function looks at the state probabilities separately for each time t, and not at the states in their sequential context.

To obtain the most probable sequence of states use `viterbi()`.

Author(s)

Rolf Turner `<r.turner@auckland.ac.nz>`

References

Rabiner, L. R., "A tutorial on hidden Markov models and selected applications in speech recognition," Proc. IEEE vol. 77, pp. 257 – 286, 1989.

See Also

`hmm()`, `rhmm()`, `viterbi()`

Examples

```
## Not run:
P <- matrix(c(0.7,0.3,0.1,0.9),2,2,byrow=TRUE)
rownames(P) <- 1:2
R <- matrix(c(0.5,0,0.1,0.1,0.3,
              0.1,0.1,0,0.3,0.5),5,2)
set.seed(42)
lll   <- sample(250:350,20,TRUE)
set.seed(909)
y.num <- rhmm(ylengths=lll,nsim=1,tpm=P,Rho=R,drop=TRUE)
fit.num <- hmm(y.num,K=2,verb=TRUE)
s.1 <- mps(y.num,fit.num)
s.2 <- mps(y.num,tpm=P,ispd=c(0.25,0.75),Rho=R)
# The order of the states has got swapped;
# note that ifelse(s.1[[1]]=="1","2","1") is much
# more similar to s.2[[1]] than is s.1[[1]].

## End(Not run)
```

`nafracCalc` *Calculate fractions of missing values.*

Description

Calculate the fraction (univariate case) or fractions (bivariate case) of missing values in the data or in each component of the data.

Usage

```
nafracCalc(y,drop=TRUE)
```

Arguments

`y`	A vector or a one or two column matrix of discrete data or a list of such vectors or matrices, or a *list* of such lists (an object of class `"multipleHmmDataSets"` such as might be produced by `rhmm()`).
`drop`	Logical scalar. If `y` is of class `"multipleHmmDataSets"` but actually consists of a single data set, and if `drop` is `TRUE`, then the returned value is not a list but rather the single component that such a list "would have had" were `drop` equal to `FALSE`. This argument is ignored if `y` is not of class `"multipleHmmDataSets"` or has length greater than 1.

Value

If `y` is *not* of class `"multipleHmmDataSets"`, then the returned value is a scalar (between 0 and 1) if the data are univariate or a pair (2-vector) of such scalars if the data are bivariate. The values are equal to the ratios of the total count of missing values in the appropriate column to the total number of observations.

If y *is* of class "multipleHmmDataSets", and if y has length greater than 1 or drop is FALSE, then the returned value is a *list* of such scalars or 2-vectors, each corresponding to one of the data sets constituting y. If y has length equal to 1 and drop is TRUE, then the returned value is the same as if codey were not of class "multipleHmmDataSets".

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

rhmm() misstify()

Examples

```
xxx <- with(SydColDisc,split(y,f=list(locn,depth)))
nafracCalc(xxx) # 0.7185199
```

---

pr *Probability of state sequences.*

Description

Calculates the conditional probability of one or more state sequences, given the corresponding observations sequences (and the model parameters.

Usage

```
pr(s, y, model=NULL, tpm, Rho, ispd=NULL, warn=TRUE)
```

Arguments

s	A sequence of states of the underlying Markov chain, or a list of such sequences or a list of lists (!!!) of such sequences.
y	A data set to which a hidden Markov model might be fitted, or a collection of such data sets in the form of an object of class "multipleHmmDataSets" as returned by rhmm() if the argument nsim is greater than 1. In this latter case s must be a list of the same length as y, and pr() is applied recursively to each pair of entries of s and y. If y consists of a single observation sequence, it is used with each of the state sequences in s in turn. Otherwise the length of the list y must be the same as the length of the list s. (If not, then an error is given). If y is missing, it is extracted from model (whence it will *not* be of class "multipleHmmDataSets"!) provided that model and its y component are not NULL. Otherwise an error is given.
model	An object of class hmm.discnp as returned by hmm().

tpm	The transition probability matrix of the chain. Ignored (and extracted from model instead) if model is not NULL.
Rho	An object specifying the "emission" probabilities of observations, given the underlying state. See hmm(). Ignored (and extracted from model instead) if model is not NULL.
ispd	The vector specifying the initial state probability distribution of the Markov chain. Ignored (and extracted from model instead) if model is not NULL. If both ispd and model are NULL then ispd is taken to be the stationary distribution of the chain, calculated from tpm.
warn	Logical scalar; should a warning be issued if Rho hasn't got relevant dimension names? (Note that if this is so, then the corresponding dimension names are formed from the sorted unique values of y or of the appropriate column(s) of y. And if *this* is so, then the user should be sure that the ordering of the entries of Rho corresponds properly to the the sorted unique values of y.) This argument is passed to the utility function check.yval() which actually issues the warning if warn=TRUE.

Value

The probability of s given y, or a vector of such probabilities if s and y are lists, or a list of such vectors if y is of class "multipleHmmDataSets".

Warning

The conditional probabilities will be tiny if the sequences involved are of any substantial length. Underflow may be a problem. The implementation of the calculations is not sophisticated.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

hmm(), mps(), viterbi(), sp(), fitted.hmm.discnp()

Examples

```
## Not run:
P <- matrix(c(0.7,0.3,0.1,0.9),2,2,byrow=TRUE)
R <- matrix(c(0.5,0,0.1,0.1,0.3,
              0.1,0.1,0,0.3,0.5),5,2)
set.seed(42)
lll   <- sample(250:350,20,TRUE)
set.seed(909)
y.num <- rhmm(ylengths=lll,nsim=1,tpm=P,Rho=R,drop=TRUE)
fit.num <- hmm(y.num,K=2,keep.y=TRUE,verb=TRUE)
# Using fitted parmeters.
s.vit.1   <- viterbi(y.num,fit.num)
pr.vit.1  <- pr(s.vit.1,model=fit.num)
```

```
# Using true parameters from which y.num was generated.
s.vit.2   <- viterbi(y.num,tpm=P,Rho=R)
pr.vit.2  <- pr(s.vit.2,y.num,tpm=P,Rho=R)
set.seed(202)
y.mult <- rhmm(fit.num,nsim=4)
s.vit.3 <- viterbi(y.mult,tpm=fit.num$tpm,Rho=fit.num$Rho)
pr.vit.3  <- pr(s.vit.3,y.mult,tpm=fit.num$tpm,Rho=fit.num$Rho)

## End(Not run)
```

`predict.hmm.discnp` *Predicted values of a discrete non-parametric hidden Markov model.*

Description

Calculates predicted values given a specification of a discrete non-parametric hidden Markov model. The specification may be provided in the form of a `hmm.discnp` object as returned by `hmm()` or in the form of "components" of such a model: the data `y`, the transition probability matrix `tpm`, the emission probabilities `Rho`, etc. If the data are numeric then these predicted values are the conditional expectations of the observations, given the entire observation sequence (and the — possibly estimated — parameters of the model). If the data are categorical (whence "expectations" make no sense) the "predicted values" are taken to be the probabilities of each of the possible values of the observations, at each time point.

Usage

```
## S3 method for class 'hmm.discnp'
predict(object, y = NULL, tpm=NULL, Rho=NULL,
                              ispd=NULL, X=NULL,addIntercept=NULL,
                              warn=TRUE, drop=TRUE, ...)
```

Arguments

`object`	If not `NULL`, an object of class `hmm.discnp` as returned by `hmm()`.
`y`	A data structure to which the fitted model `object` *could* have been fitted. If `y` is NULL, an attempt is made to extract `y` from `model`.
`tpm`, `Rho`, `ispd`, `X`, `addIntercept`, `warn`	See the help for `sp()`.
`drop`	Logical scalar. See the help for `fitted.hmm.discnp()`.
`...`	Not used.

Details

This function is essentially the same as `fitted.hmm.discnp()`. The main difference is that it allows the calculation of fitted/predicted values for a data object `y` possibly different from that to which the model was fitted. Note that if both the argument `y` and `object[["y"]]` are present, the "argument" value takes precedence. This function also allows the model to be specfied in terms of individual components rather than as a fitted model of class `"hmm.discnp"`. These components, (`tpm`, `Rho`, `ispd`, `X`, `addIntercept`) if supplied, *take precedence* over the corresponding components of `object`. The opposite applies with `sp()`. The function `fitted.hmm.discnp()` makes use *only* of the components of `object`.

Value

See the help for `fitted.hmm.discnp()`.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

`sp()` `link{fitted.hmm.discnp}()`

Examples

```
P <- matrix(c(0.7,0.3,0.1,0.9),2,2,byrow=TRUE)
R <- matrix(c(0.5,0,0.1,0.1,0.3,
              0.1,0.1,0,0.3,0.5),5,2)
set.seed(42)
ll1  <- sample(250:350,20,TRUE)
y1   <- rhmm(ylengths=ll1,nsim=1,tpm=P,Rho=R,drop=TRUE)
fit  <- hmm(y1,K=2,verb=TRUE,keep.y=TRUE,itmax=10)
fv   <- fitted(fit)
set.seed(176)
ll2  <- sample(250:350,20,TRUE)
y2   <- rhmm(ylengths=ll2,nsim=1,tpm=P,Rho=R,drop=TRUE)
pv   <- predict(fit,y=y2)
yval <- letters[1:5]
set.seed(171)
y3   <- rhmm(ylengths=ll2,yval=yval,nsim=1,tpm=P,Rho=R,drop=TRUE)
fit3 <- hmm(y3,K=2,verb=TRUE,keep.y=TRUE,itmax=10)
pv3  <- predict(fit3) # Same as fitted(fit3).
```

`rhmm` *Simulate discrete data from a non-parametric hidden Markov model.*

Description

Simulates one or more replicates of discrete data from a model such as is fitted by the function `hmm()`.

Usage

```
rhmm(model,...,nsim,verbose=FALSE)
## Default S3 method:
rhmm(model, ..., nsim=1, verbose=FALSE, ylengths,
                        nafrac=NULL, fep=NULL, tpm, Rho, ispd=NULL, yval=NULL,
                        drop=TRUE, forceNumeric=TRUE)
## S3 method for class 'hmm.discnp'
rhmm(model, ..., nsim=1, verbose=FALSE, inMiss=TRUE,
                           fep=NULL, drop=TRUE, forceNumeric=TRUE)
```

Arguments

`model`	An object of class `hmm.discnp`. This will have the form of a list specifying a hidden Markov model with discrete emissions and emission probabilities specified non-parametrically, i.e. by means of some form of table or tables. Usually this will be an object returned by `hmm()`. This argument is ignored by the default method.
`...`	Not used.
`nsim`	Integer scalar; the number of data sets to be simulated.
`verbose`	Logical scalar. If `TRUE` then the overall index of the simulated value that has been reached is printed out every 1000 iterations. Useful for reassurance when very "large" simulations are undertaken.
`ylengths`	Integer values vector specify the lengths (or number of rows in the bivariate setting) of the individual observation sequences constituting a data set.
`nafrac`	See `misstify()` for an explanation of this argument. If specified a fraction `nafrac[[j]]` of column `j` of the data will be randomly set equal to `NA`.
`fep`	"First entry present". See `misstify()` for an explanation of this argument.
`tpm`	The transition probability matrix for the underlying hidden Markov chain(s). Note that the rows of `tpm` must sum to 1. Ignored if `ncol(Rho)==1`. Ignored by the `hmm.discnp` method and extracted from `model`.
`Rho`	An object specifying the probability distribution of the observations, given the state of the underlying hidden Markov chain. (I.e. the "emission" probabilities.) See `hmm()`. Note that `Rho` can be such that the number of states is 1, in which case the simulated data are i.i.d. from the single distribution specified by `Rho`. Ignored by the `hmm.discnp` method and extracted from `model`.
`ispd`	A vector specifying the initial state probability distribution of the chain. If this is not specified it is taken to be the stationary distribution of the chain, calculated from `tpm`. Ignored by the `hmm.discnp` method and extracted from `model`.
`yval`	Vector of possible values of the observations, or (in the bivariate setting) a list of two such vectors. If not supplied it is formed from the levels of the factor

constituting the y column of Rho (univariate case) or from appropriate dimension names associated with Rho (bivariate case). Ignored by the hmm.discnp method.

drop — Logical scalar; if TRUE then lists of length 1 are replaced by their first entry. In particular if nsim is 1 and if drop is TRUE then the list to be returned by this function (see below) is replaced by its first and only entry. Also if ylengths is of length 1 (so that each entry of the returned value contains only a single sequence of of simulated observations) then each list of such sequences is replaced by its first and only entry.

inMiss — Logical scalar; if TRUE then missing values will be randomly inserted into the data in the fraction nafrac determined from object.

forceNumeric — Logical scalar; if TRUE then if all of the possible values of the observations can be interpreted as numeric (by as.numeric()) then they are so interpreted. That is, the value returned will consist of a collection of numeric sequences, rather than a collection of sequences of values of categorical variables.

Value

If nsim>1 or drop is FALSE then the value returned is a list of length nsim. Each entry of this list is in turn a list of the same length as ylengths, each component of which is an independent vector or matrix of simulated observations. The length or number of rows of component i of this list is equal to ylengths[i]. The values of the observations are entries of yval or of *its* entries when yval is a list.

If nsim=1 and drop is TRUE then the ("outer") list described above is replaced by its first and only entry

If the length of ylengths is 1 and drop is TRUE then each "inner" list described above is replaced by its first and only entry.

Note

You may find it useful to avail yourself of the function nafracCalc() to determine the fraction of missing values in a given existing (presumably "real") data set.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

hmm() nafracCalc() misstify()

Examples

```
# To do: one or more bivariate examples.
## Not run:
    y <- list(linLandFlows$deciles,ftLiardFlows$deciles)
    fit <- hmm(y,K=3)
    simX <- rhmm(fit)
```

```
## End(Not run)
```

---

scovmat *Simulation based covariance matrix.*

Description

Produces an estimate of the covariance matrix of the parameter estimates in a model fitted by hmm.discnp. Uses a method based on simulation (or "parametric bootstrapping").

Usage

```
scovmat(object, expForm=TRUE, seed = NULL, nsim=100, verbose = TRUE)
```

Arguments

object	An object of class hmm.discnp as returned by hmm().
expForm	Logical scalar. Should the covariance matrix produced be that of the estimates of the parameters expressed in "exponential" (or "smooth" or "logistic") form? If expForm=FALSE then the parameter estimates considered are "raw" probabilities, with redundancies (last column of tpm; last row of Rho) removed.
seed	Integer scalar serving as a seed for the random number generator. If left NULL the seed itself is chosen randomly from the set of integers between 1 and 10^5.
nsim	A positive integer. The number of simulations upon which the covariance matrix estimate will be based.
verbose	Logical scalar; if TRUE, iteration counts will be printed out during each of the simulation and model-fitting stages.

Details

This function is currently applicable only to models fitted to univariate data. If there are *predictors* in the model, then only the exponential form of the parameters may be used, i.e. expForm *must* be TRUE.

Value

A (positive definite) matrix which is an estimate of the covariance of the parameter estimates from the fitted model specified by object. It has row and column labels which indicate the parameters to which its entries pertain, in a reasonably perspicuous manner.

This matrix has an attribute seed (the random number generation seed that was used) so that the calculations can be reproduced.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

squantCI() link{rhmm}() link{hmm)}()

Examples

```
## Not run:
y   <- list(lindLandFlows$deciles,ftLiardFlows$deciles)
fit <- hmm(y,K=3)
ccc <- scovmat(fit,nsim=100)

## End(Not run)
```

---

sp *Calculate the conditional state probabilities.*

Description

Returns the probabilities that the underlying hidden state is equal to each of the possible state values, at each time point, given the observation sequence.

Usage

```
sp(y, model = NULL, tpm=NULL, Rho=NULL, ispd=NULL, X=NULL,
   addIntercept=NULL, warn=TRUE, drop=TRUE)
```

Arguments

y	The observations on the basis of which the probabilities of the underlying hidden states are to be calculated. May be a vector of a one or two column matrix of observations, or a list each component of which is such a vector or matrix. If y is missing it is set equal to the y component of model, given that that argument is non-NULL and that that component exists. Otherwise an error is given.
model	An object of class hmm.discnp as returned by hmm().
tpm	The transition probability matrix for the underlying hidden Markov chain. Ignored if model is not NULL (in which case tpm is extracted from model).
Rho	An object specifying the distribution of the observations, given the underlying state. I.e. the "emission" probabilities. See hmm(). Ignored if model is not NULL (in which case Rho is extracted from model).
ispd	Vector specifying the initial state probability distribution of the underlying hidden Markov chain. Ignored if model is not NULL (in which case ispd is extracted from model). If both model[["ispd"]] and ispd are NULL then ispd is calculated to be the stationary distribution of the chain as determined by tpm.
X	An optional *numeric* matrix, or a list of such matrices, of *predictors*. Ignored if model is not NULL (in which case X is extracted from model). The use of such predictors is (currently, at least) applicable only in the univariate emissions setting. If X is a list it must be of the same length as y and all entries

of this list must have the same number of columns. The number of rows of each entry must be equal to the length of the corresponding entry of y. If X is a matrix then y should be a vector or one-column matrix (or a list with a single entry equal to such).

addIntercept — Logical scalar. See the documentation of hmm(). Ignored if model is not NULL (in which case addIntercept is extracted from model).

warn — Logical scalar; should a warning be issued if Rho hasn't got relevant dimension names? (Note that if this is so, then the corresponding dimension names are formed from the sorted unique values of y or of the appropriate column(s) of y. And if *this* is so, then the user should be sure that the ordering of the entries of Rho corresponds properly to the the sorted unique values of y.) This argument is passed to the utility function check.yval() which actually issues the warning if warn=TRUE.

drop — Logical scalar. If y is a matrix, or a list of length 1, and if drop is FALSE then the returned value is a list whose sole entry is the matrix that would have been returned were drop equal to TRUE. The argument drop is ignored if y is a list of length greater than 1.

Details

Note that in contrast to predict.hmm.discnp(), components in model take precendence over individually supplied components (tpm, Rho, ispd, X and addIntercept).

Value

If y is a single matrix of observations or a list of length 1, and if drop is TRUE then the returned value is a matrix whose rows correspond to the states of the hidden Markov chain, and whose columns correspond to the observation times. Otherwise the returned value is a list of such matrices, one for each matrix of observations.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

hmm(), mps(), viterbi(), pr(), fitted.hmm.discnp()

Examples

```
P <- matrix(c(0.7,0.3,0.1,0.9),2,2,byrow=TRUE)
R <- matrix(c(0.5,0,0.1,0.1,0.3,
              0.1,0.1,0,0.3,0.5),5,2)
set.seed(42)
y    <- rhmm(ylengths=rep(300,20),nsim=1,tpm=P,Rho=R,drop=TRUE)
fit  <- hmm(y,K=2,verb=TRUE,keep.y=TRUE,itmax=10)
cpe1 <- sp(model=fit) # Using the estimated parameters.
cpe2 <- sp(y,tpm=P,Rho=R,warn=FALSE) # Using the ``true'' parameters.
```

```
# The foregoing would issue a warning that Rho had no row names
# were it not for the fact that "warn" has been set to FALSE.
```

---

squantCI *Simulation-quantile based confidence intervals.*

Description

Calculates estimates of confidence intervals for the parameters of a model fitted by hmm.discnp. Uses a method based quantiles of estimates produced by simulation (or "parametric bootstrapping").

Usage

```
squantCI(object, expForm = TRUE, seed = NULL, alpha = 0.05,
         nsim=100, verbose = TRUE)
```

Arguments

object	An object of class hmm.discnp as returned by hmm().
expForm	Logical scalar. Should the confidence intervals produced be for the parameters expressed in "exponential" (or "smooth" or "logistic") form? If expForm=FALSE then the parameters considered are "raw" probabilities, with redundancies (last column of tpm; last row of Rho) removed.
seed	Integer scalar serving as a seed for the random number generator. If left NULL the seed itself is chosen randomly from the set of integers between 1 and 10^5.
alpha	Positive real number strictly between 0 and 1. A set of 100*(1-alpha)% confidence intervals will be produced.
nsim	A positive integer. The number of simulations upon which the confidence interval estimates will be based.
verbose	Logical scalar; if TRUE, iteration counts will be printed out during each of the simulation and model-fitting stages.

Details

This function is currently applicable only to models fitted to univariate data. If there are *predictors* in the model, then only the exponential form of the parameters may be used, i.e. expForm *must* be TRUE.

Value

A npar-by-2 matrix (where npar is the number of "independent" parameters in the model) whose rows form the estimated confidence intervals. (The first entry of each row is the lower bound of a confidence interval for the corresponding parameter, and the second entry is the upper bound. The row labels indicate the parameters to which each row pertains, in a reasonably perspicuous manner. The column labels indicate the relevant quantiles in percentages.

This matrix has an attribute seed (the random number generation seed that was used) so that the calculations can be reproduced.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

scovmat() link{rhmm}() link{hmm)}()

Examples

```
## Not run:
y   <- list(lindLandFlows$deciles,ftLiardFlows$deciles)
fit <- hmm(y,K=3)
CIs <- squantCI(fit,nsim=100)

## End(Not run)
```

SydColDisc *Discretised version of coliform counts in sea-water samples*

Description

Discretised version of counts of faecal coliform bacteria in sea water samples collected at seven locations near Sydney NSW, Australia. There were four "controls": Longreef, Bondi East, Port Hacking "50", and Port Hacking "100" and three "outfalls": Bondi Offshore, Malabar Offshore and North Head Offshore. At each location measurements were made at four depths: 0, 20, 40, and 60 meters. A large fraction of the counts are missing values.

Usage

```
SydColDisc
```

Format

A data frame with 5432 observations on the following 6 variables.

`y` A factor consisting of a discretisation of counts of faecal coliform count bacteria in sea water samples. The original measures were obtained by a repeated dilution process. The data were discretised using the cut() function with breaks given by `c(0,1,5,25,200,Inf)` and labels equal to `c("lo","mlo","m","mhi","hi")`.

`locn` a factor with levels `Longreef`, `Bondi East`, `Port Hacking 50`, `Port Hacking 100`, `Bondi Offshore`, `Malabar Offshore` and `North Head Offshore`.

`depth` a factor with levels `0` (0 metres), `20` (20 metres), `40` (40 metres), `60` (60 metres).

`ma.com` A factor with levels `no` and `yes`, indicating whether the Malabar sewage outfall had been commissioned.

`nh.com` A factor with levels `no` and `yes`, indicating whether the North Head sewage outfall had been commissioned.

`bo.com` A factor with levels `no` and `yes`, indicating whether the Bondi Offshore sewage outfall had been commissioned.

Details

The observations corresponding to each location-depth combination constitute a (discrete valued) time series. The sampling interval is ostensibly 1 week; distinct time series are ostensibly synchronous. The measurements were made over a 194 week period. Due to exigencies of weather, the unreliabitity of boats and other factors the collection times were actually highly irregular and have been rounded to the neares week. Often no sample was obtained at a given site within a week of the putative collection time, in which the observed count is given as a missing value. In fact over **75%** of the counts are missing. See Turner et al. (1998) for more detail.

Modelling

The hidden Markov models applied in the paper Turner et al. (1998) and in the paper Turner (2008) used a numeric version of the response in this data set. The numeric response was essentially a square root transformation of the original data, and the resulting values were modelled in terms of a Poisson distribution. See the references for details.

Source

The original data were kindly supplied by Geoff Coade, of the New South Wales Environment Protection Authority (Australia)

References

T. Rolf Turner, Murray A. Cameron, and Peter J. Thomson. Hidden Markov chains in generalized linear models. *Canadian J. Statist.* **26** (1998) 107 – 125.

Rolf Turner. Direct maximization of the likelihood of a hidden Markov model. *Comp. Statist. Data Anal.* **52** (2008) 4147–4160.

---

update.hmm.discnp *Update a fitted* hmm.discnp *model.*

Description

An update() method for objects of class hmm.discnp.

Usage

```
## S3 method for class 'hmm.discnp'
update(object,..., data, Kplus1=FALSE,
        tpm2=NULL, verbose=FALSE, method=NULL, optimiser=NULL,
        stationary=NULL, mixture=NULL, cis=NULL, tolerance=NULL,
        itmax=NULL, crit=NULL, X=NULL, addIntercept=NULL)
```

Arguments

object	An object of class hmm.discnp as returned by hmm().
...	Not used.
data	The data set to which the (updated) model is to be fitted. See the description of the y argument of hmm() for more detail.
Kplus1	Logical scalar. Should the number of states be incremented by 1? If so then tpm (the transition probability matrix) is re-formed by rbind()-ing on a row all of whose entries are 1/K (where K is the "old" number of states) and then cbind()-ing on a column of zeroes. The emission probability matrix Rho is reformed by cbind()-ing on a column all of whose entries are 1/m where m is the number of discrete values of the emissions. Note that the intial likelihood of the "new" model with K+1 states will (should?) be exactly the same as that of of the "old" fitted K-state model. The Kplus1 argument is provided mainly so as to provide a set of starting values for the fitting process which will guarantee the log likelihood of a K+1-state model will be at least as large as that of a K-state model fitted to the same data set. Experience indicates that when Kplus1=TRUE is used, the fitting process does not "move very far" from the maximum log likelihood found for the K-state model. It is then advisable to try (many) random starting values so as to (try to) find the "true" maximum for the K+1-state model.
tpm2	The transtion probability matrix to use when updating a model fitted with K=1 and Kplus1=TRUE. This argument is ignored otherwise. The default value of this argument is matrix(0.5,2,2). The value of tpm2 makes no difference to the *initial* value of the likelihood of the K=2 model (which will be identical to the likelihood of the fitted K=1 model that is being updated). Any two-by-two transition probability matrix "will do". However the value of tpm2 could conceivably have an impact on the final likelihood of the K=2 model to which the fitting procedure converges. This is particularly true if the method is (or is switched to) "LM".
verbose	See the help for hmm().
method	See the help for hmm().
optimiser	See the help for hmm().
stationary	See the help for hmm().
mixture	See the help for hmm().
cis	See the help for hmm().
tolerance	See the help for hmm().
itmax	See the help for hmm().
crit	See the help for hmm().
X	See the help for hmm().
addIntercept	See the help for hmm().

Details

Except for argument X, any arguments that are left NULL have their values supplied from the args component of object.

Value

An object of class hmm.discnp with an additional component init.log.like which is the initial log likelihood calculated at the starting values of the parameters (which may be modified from the parameters returned in the object being updated, if Kplus1 is TRUE). The calculation is done by the function logLikHmm(). Barring the strange and unforeseen, init.log.like should be (reassuringly) equal to object$log.like. See hmm() for details of the other components of the returned value.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

See Also

hmm() rhmm.hmm.discnp()

Examples

```
set.seed(294)
fit  <- hmm(WoodPeweeSong,K=2,rand.start=list(tpm=TRUE,Rho=TRUE),itmax=10)
xxx  <- rhmm(fit,nsim=1)
sfit <- update(fit,data=xxx,itmax=10)
yyy  <- with(SydColDisc,split(y,f=list(locn,depth)))
f1   <- hmm(yyy,K=1)
f2   <- update(f1,data=yyy,Kplus1=TRUE) # Big improvement, but ...
## Not run:
g2   <- hmm(yyy,K=2) # Substantially better than f2.

## End(Not run)
```

---

viterbi *Most probable state sequence.*

Description

Calculates "the" most probable state sequence underlying each of one or more replicate observation sequences.

Usage

```
viterbi(y, model = NULL, tpm, Rho, ispd=NULL,log=FALSE, warn=TRUE)
```

Arguments

y	The observations for which the most probable sequence(s) of underlying hidden states are required. May be a sequence of observations in the form of a vector or a one or two column matrix, or a list each component of which constitutes a (replicate) sequence of observations. It may also be an object of class "multipleHmmDataSets" as returned by rhmm() with nsim>1. If y is missing, it is extracted from model (whence it will *not* be of class "multipleHmmDataSets"!) provided that model and its y component are not NULL. Otherwise an error is given.
model	An object describing a hidden Markov model, as fitted to the data set y by hmm().
tpm	The transition probability matrix for a hidden Markov model; ignored if model is non-null.
Rho	An object specifying the probability distributions of the observations for a hidden Markov model. See hmm(). Ignored if model is non-null. Should bear some reasonable relationship to y. If Rho has dimension names (or if its entries have dimension names in the case where Rho is a list) then the appropriate dimension names must include all corresponding values of the observations. If a relevant vector of dimension names is NULL then it is formed as the sort unique values of the approprate columns of the observation matrices. In this case the corresponding dimensions must match the number of unique values.
ispd	The initial state probability distribution for a hidden Markov model; ignored if model is non-null. Should bear some reasonable relationship to y. If model and ispd are both NULL then ispd is set equal to the stationary distribution calculated from tpm.
log	Logical scalar. Should logarithms be used in the recursive calculations of the probabilities involved in the Viterbi algorithm, so as to avoid underflow? If log is FALSE then underflow is avoided instead by a normalization procedure. The quantity delta (see Rabiner 1989, page 264) is replaced by delta/sum(delta) at each step. It should actually make no difference whether log is set to TRUE. I just included the option because I could. Also the HMM package uses the logarithm approach so setting log=TRUE might be of interest if comparisons are to be made between results from the two packages.
warn	Logical scalar; should a warning be issued if Rho hasn't got relevant dimension names? (Note that if this is so, then the corresponding dimension names are formed from the sorted unique values of y or of the appropriate column(s) of y. And if *this* is so, then the user should be sure that the ordering of the entries of Rho corresponds properly to the the sorted unique values of y.) This argument is passed to the utility function check.yval() which actually issues the warning if warn=TRUE.

Details

Applies the Viterbi algorithm to calculate "the" most probable robable state sequence underlying each observation sequences.

Value

If y consists of a single observation sequence, the value is the underlying most probable observation sequence, or a matrix whose columns consist of such sequences if there is more than one (equally) most probable sequence.

If y consists of a list of observation sequences, the value is a list each entry of which is of the form described above.

If y is of class "multipleHmmDataSets" then the value returned is a list of lists of the sort described above.

Warning

There *may* be more than one equally most probable state sequence underlying a given observation sequence. This phenomenon can occur but appears to be unlikely to do so in practice.

Thanks

The correction made to the code so as to avoid underflow problems was made due to an inquiry and suggestion from Owen Marshall.

Author(s)

Rolf Turner <r.turner@auckland.ac.nz>

References

Rabiner, L. R., "A tutorial on hidden Markov models and selected applications in speech recognition," Proc. IEEE vol. 77, pp. 257 – 286, 1989.

See Also

hmm(), rhmm(), mps(), pr()

Examples

```
# See the help for logLikHmm() for how to generate y.num and y.let.
## Not run:
fit.num     <- hmm(y.num,K=2,verb=TRUE,keep.y=TRUE)
v.1         <- viterbi(model=fit.num)
rownames(R) <- 1:5 # Avoids a (harmless) warning.
v.2         <- viterbi(y.num,tpm=P,Rho=R)
# P and R as in the help for logLikHmm() and for sp().

# Note that the order of the states has gotten swapped; 3-v.1[[1]]
# is identical to v.2[[1]]; for other k = 2, ..., 20, 3-v.1[[k]]
# is much more similar to v.2[[k]] than is v.1[[k]].

fit.let <- hmm(y.let,K=2,verb=TRUE,keep.y=TRUE))
v.3 <- viterbi(model=fit.let)
rownames(R) <- letters[1:5]
```

```
v.4 <- viterbi(y.let,tpm=P,Rho=R)

## End(Not run)
```

`weissData` *Data from "An Introduction to Discrete-Valued Time Series"*

Description

Data sets from the book "An Introduction to Discrete-Valued Time Series" by Christian H. Weiß.

Usage

```
data(Bovine)
data(Cryptosporidiosis)
data(Downloads)
data(EricssonB_Jul2)
data(FattyLiver)
data(FattyLiver2)
data(goldparticle380)
data(Hanta)
data(InfantEEGsleepstates)
data(IPs)
data(LegionnairesDisease)
data(OffshoreRigcountsAlaska)
data(PriceStability)
data(Strikes)
data(WoodPeweeSong)
```

Format

- `Bovine` A character vector of length 8419.
- `Cryptosporidiosis` A numeric (integer) vector of length 365.
- `Downloads` A numeric (integer) vector of length 267.
- `EricssonB_Jul2` A numeric (integer) vector of length 460.
- `FattyLiver2` A numeric (integer) vector of length 449.
- `FattyLiver` A numeric (integer) vector of length 928.
- `goldparticle380` A numeric (integer) vector of length 380.
- `Hanta` A numeric (integer) vector of length 52.
- `InfantEEGsleepstates` A character vector of length 107.
- `IPs` A numeric (integer) vector of length 241.
- `LegionnairesDisease` A numeric (integer) vector of length 365.
- `OffshoreRigcountsAlaska` A numeric (integer) vector of length 417.
- `PriceStability` A numeric (integer) vector of length 152.
- `Strikes` A numeric (integer) vector of length 108.
- `WoodPeweeSong` A numeric (integer) vector of length 1327.

Details

For detailed information about each of these data sets, see the book cited in the **References**.

Note that the data sets `Cryptosporidiosis` and `LegionnairesDisease` are actually called `Cryptosporidiosis_02-08` and `LegionnairesDisease_02-08` in the given reference. The "suffixes" were removed since the minus sign causes problems in a variable name in `R`.

Source

These data sets were kindly provided by Prof. Christian H. Weiß. The package author is also pleased to acknowledge the kind permission granted by Prof. Kurt Brännäs (Professor Emeritus of Economics at Umeå University) to include the Ericsson time series data set (`EricssonB_Jul2`).

References

Christian H. Weiß (2018). *An Introduction to Discrete-Valued Time Series*. Chichester: John Wiley & Sons.

Examples

```
## Not run:
fit1 <- hmm(WoodPeweeSong,K=2,verbose=TRUE)
# EM converges in 6 steps --- suspicious.
set.seed(321)
fit2 <- hmm(WoodPeweeSong,K=2,verbose=TRUE,rand.start=list(tpm=TRUE,Rho=TRUE))
# 52 steps --- note the huge difference between fit1$log.like and fit2$log.like!
set.seed(321)
fit3 <- hmm(WoodPeweeSong,K=2,verbose=TRUE,method="bf",
            rand.start=list(tpm=TRUE,Rho=TRUE))
# log likelihood essentially the same as for fit2

## End(Not run)
```

Index