Package 'eglhmm'

May 8, 2026

Version 0.1-3

Date 2024-02-13

Title Extended Generalised Linear Hidden Markov Models

Maintainer Rolf Turner <rolfturner@posteo.net>

Description Fits a variety of hidden Markov models, structured in an extended generalized linear model framework. See T. Rolf Turner, Murray A. Cameron, and Peter J. Thomson (1998) <doi:10.2307/3315677>, and Rolf Turner (2008) <doi:10.1016/j.csda.2008.01.029> and the references cited therein.

Imports dbd, nnet

Suggests R.rsp

VignetteBuilder R.rsp

LazyData true

ByteCompile true

NeedsCompilation yes

Depends R (>= 3.5)

License GPL (>= 2)

Author Rolf Turner [aut, cre]

Repository CRAN

Date/Publication 2024-02-12 23:40:02 UTC

Contents

`anova.eglhmm` *Test for a difference between two fitted* `eglhmm` *models.*

Description

Apply a likelihood ratio test to determine whether the difference, between the log likelihood statistics of two fitted `eglhmm` models, is statistically significant.

Usage

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

Arguments

`object`	An object of class `"eglhmm"` as returned by `eglhmm()`.
`...`	Precisely *one* more object of class `"eglhmm"`, to be compared with `object`.

Details

This anova method handles only comparisons between *two* models.) The order of the arguments (i.e. which object is passed as "`object`" and which is passed as the sole entry of the ... argument) is immaterial.

Value

A list with components

- `stat` the likelihood ratio statistic, i.e. the difference between the log likelihoods of the two models. That for the model with the smaller number of parameters is subtracted from that for the model with the larger number.
- `df` the degrees of freedom of the likelihood ratio statistic, i.e. the difference between the number of parameters of the respective models. The smaller number is subtracted from the larger.
- `pvalue` the p-value of the test as given by `pchisq(stat, df, lower.tail = FALSE)`.

This list has an attribute `"details"` consisting of a numeric vector of length four with entries `ll1` (the smaller of the log likelihoods), `ll2` (the larger of the log likelihoods), `np1` (the smaller of the parameter counts) and `np2` (the larger of the parameter counts).

Author(s)

Rolf Turner <rolfturner@posteo.net>

Examples

```
fit1 <- eglhmm(formula=y~locn+depth,data=SydColCount,
               cells=c("locn","depth"),distr="P",K=2,
               method="em",verb=TRUE)
fit2 <- eglhmm(formula=y~locn+depth+ma.com+nh.com+bo.com,data=SydColCount,
               cells=c("locn","depth"),distr="P",K=2,
               method="em",verb=TRUE)
anova(fit1,fit2)
```

---

bcov *Bootstrap covariance.*

Description

Creates an estimate of the covariance matrix of the parameter estimates for an extended generalised linear hidden Markov model via parametric bootstrapping.

Usage

```
bcov(object, nsim = 50, itmax = 500, verbose = TRUE)
```

Arguments

object	An object of class eglhmm as produced by eglhmm().
nsim	The number of data sets to simulate, from which to estimate parameters. From each data set a vector of parameters is estimated; the estimated covariance matrix is the empirical covariance matrix of these nsim vectors.
itmax	The maximum number of iterations to be used in attempting to achieve convergence when fitting models to the simulated data sets. Note that if convergence is not achieved, the simulated data set being used is discarded (i.e. it "doesn't count") and a replacement data set is simulated.
verbose	Logical scalar. Should a "progress report" be printed out at each step of the fitting procedure?

Value

A list with components:

C_hat	The parametric bootstrap estimate of the covariance matrix of the parameter estimates.
nc.count	A count of the total number of times that the algorithm failed to converge during the bootstrapping procedure.
an.count	A count of the "anomalies" that occurred, i.e. the number of times that there was a decrease in the log likelihood. Present only if the method used in fitting the models is "em".

Remarks

Although this documentation refers to "extended generalised linear models", the only such models currently (13/02/2024) available are the Gaussian model with the identity link, the Poisson model, with the log link, the Binomial model with the logit link, the Dbd (discretised beta distribution model), and the Multinom model. The latter two are generalised linear models only in the "extended" sense. Other models may be added at a future date.

When `eglhmm()` is called by `bcov()` the argument `checkDecrLL` is set equal to `FALSE`. This has an effect only when the `method` used in fitting the models is `"em"`. In this case a decrease in the log likelihood is treated as meaning that the algorithm has converged. Setting `checkDecrLL` equal to `FALSE` is done so as to decrease the number of discarded data sets and thereby speed up the rate at which the iterations proceed.

Author(s)

Rolf Turner <rolfturner@posteo.net>

References

See the help for `eglhmm()` for references.

See Also

`fitted.eglhmm()` `reglhmm()` `reglhmm.default()` `reglhmm.eglhmm()`

Examples

```
    ## Not run:  # Takes too long.
        fitP <- eglhmm(y~locn+depth,data=SydColCount,distr="P",cells=c("locn","depth"),
                    K=2,contr="sum",verb=TRUE,itmax=300)
cvrP <- bcov(fitP)
        fitD <- eglhmm(y~locn+depth,data=SydColCount,distr="D",cells=c("locn","depth"),
                    K=2,nbot=0,ntop=11,contr="sum",verb=TRUE)
cvrD <- bcov(fitD)
        fitM <- eglhmm(y~locn+depth,data=SydColDisc,distr="M",cells=c("locn","depth"),
                    K=2,contr="sum",verb=TRUE)
cvrM <- bcov(fitM)

## End(Not run)
```

`crossval` *Cross validate an extended generalised linear hidden Markov model.*

Description

Calculates a number of cross validation log likelihood values for a hidden Markov model (usually one fitted by the `eglhmm()` function).

Usage

```
crossval(model,data,nrep,frac=0.8,type,id="id",minNcomp=100,
         seed=NULL,crossVerb=FALSE,lastPar=NULL,...)
```

Arguments

model	A list having components with names selected from those of objects returned by `eglhmm()` e.g. `distr`, `theta`, `formula` etc. Typically `model` will be of class `"eglhmm"` and will have been returned by the `eglhmm()` function.
data	A data frame containing the observations to which the cross validation are to be fitted. It is of course up to the user to ensure that a specified value of `data` makes sense, i.e. is consistent with the other arguments.
nrep	Positive nteger scalar; the number of replications, i.e. the number of cross validation calculations undertaken. If argument `lastPar` (see below) is supplied then `nrep` is ignored and is silently set equal to 1. If `lastPar` is `NULL` then `nrep` must be supplied, otherwise an error is thrown.
frac	Postive scaler, less than 1. The fraction of the data randomly selected to be used as `training data` on each replication. (The remaining data, i.e. those data *not* used as training data, are used as `validation` data.
type	Integer scalar, equal to either 1 or 2, which determines the nature of the sampling used to produce the training and validation data. If `type=1` then these data sets are obtained by sampling data points individually. The training data are obtained by setting a fraction `1 - frac` of the observed emission values (those which are *not* missing already) equal to `NA`. The validation data are the complement of the training data. If `type=2` then the *components* of `data` are randomly sampled (and used in their entirety, either for training or for validation). The components are determined from the column of `data` the name of which is specified by the argument `id`; if there is no such column, then an error is thrown. Sampling the components means sampling the levels of (the factor) `data[,id]`. Obviously it is sensible to use `type=2` *only* if `data` has a *large* number of components. By default this number is required to be at least 100. (See `minNcomp` below.)
id	Character scalar specifying the column to be used to determine the individual independent time series that make up the data. Ignored unless `type=2`.
minNcomp	Integer scalar specifying the minimum number of components (number of levels of `data["id"]`) that `data` must have in order for `type=2` to used. Ignored if `type` is equal to 1. If the number of components is less than the default value of `minNcomp` (i.e. 100) then it is strongly recommended that `type=1` be used instead.
seed	Positive integer scalar to be used as a seed for the random number generator (so as to enable reproducibility). If not supplied, it is randomly chosen from the sequence `1:1e6`. Note that if `nrep > 1` then after this seed is set, a vector `SEEDS` of "auxiliary" seeds, of length `nrep`, is chosen from the sequence `1:1e6` and

	the seed is set from the corresponding entry of this vector at the start of each replication. If `nrep==1` then the sampling for the single replication that occurs is determined by `seed`.
`crossVerb`	Logical scalar. Should brief "progress reports" (letting the user know what is happening with respect to replicate `repl`, for each `repl`) be produced?
`lastPar`	The last values of the (relevant) fitted paramenters, provided as an attribute of a component of the list returned by the function currently under consideration (i.e. `crossval()`, whenever the process of fitting the model in question to the training data did not converge. These values can be used as starting values so as to carry on with the fitting process from where the previous attempt left off.
`...`	Possible additional arguments to be passed to `eglhmm()` via `update()`, e.g. `itmax`, `tolerance`, ...

Details

On each replication a random subset comprising `frac` of the data is selected to serve as training data. The complement of this subset is used as validation data. The model specified by `model` is fitted to the training data. It is possible to over-ride some of the details of the specifications producing `model`, via the `...` argument of `crossval()`. After the model is fitted to the training data, the log likelihood of the validation data is calculated on the basis of that fitted model.

If the procedure for fitting a model to the training data fails to converge, then the corresponding entry of the list returned by this function is `NA`. In this case, the entry is assigned an attribute `lastPar`, (the estimates of the model parameters that were current when the fitting algorithm terminated) which will in turn have attributes `trnDat`, `valDat` (the training data in question and the corresponding validation data), and `seed` (the value of the seed that was set before the sampling that determined `trnDat` and `valDat` took place). The value of `seed` is `SEEDS[i]` (if `nrep>1` and the entry in question was the `i`th entry of the returned list) or the value of the `seed` argument of this function or its random replacement if this argument was not supplied (if `nrep==1`).

The attribute `lastPar` enables the user to continue the procedure for fitting a model to the training data, starting from where the procedure, that failed to converge, left off. Continuing the procedure is easily effected by calling `crossval()` with argument `par0` set equal to the `lastPar` attribute of the relevant entry of the list that was previously returned by this function.

If `type==1` then the training and validation data are created in a somewhat subtle manner. The procedure necessitates referring to the "original" data. The data frame that is passed to `eglhmm()` is a "replicated" version of the original data, with each row of the original data being repeated once for each level of `state` (and a `"state"` column — factor — being added to the resulting data frame). If `type==2` the procedure is conceptually simpler. The procedure in the `type==1` setting is as follows:

- Let S be the set of indices of all non-missing values in the column of the original data that contains the emissions.
- Select at random a subset V of S so that the size of V is the fraction `frac` of the size of S.
- Let T be the complement in S of V.
- The training data are formed by replacing by `NA` all those values of the emissions column in the original data whose indices are in T.
- The validation data are formed by replacing by `NA` all those values of the emissions column in the original data, whose indices are in V.

- Then replicate both the training and validation data in the manner described above.

  If `type==2` then the training data are formed by selecting at random a fraction `frac` of the levels of the column of `data` named `"id"`. (If there is no such column, an error is thrown.) The training data then consist of those rows of `data` corresponding to the selected levels of `id`. The validation data then consist of those rows of `data` which are not in the training data.

Value

If `nrep>1` the returned value is list of length `nrep`. The `i`th entry of this list is the log likelihood of the validation data with respect to the model fitted to the training data, for the `i`th random selection of these two subsets. This entry will be `NA` if the attempt to fit a model to the training data was unsuccessful. The `i`th entry has an attribute `seed` (singular) which is the value of the seed that was set prior to the random sampling that chose the training and validation data. If the `i`th entry is `NA` it will also have an attribute `lastPar` which in turn will have attributes `trnDat` and `valDat`. See **Details**.

If `nrep>1` then the returned value also has an attribute `seeds` (plural) which is vector of length `nrep+1`, consisting of the "auxiliary" seed vector `SEEDS` (see the argument `seed`) together with the "over all" seed (possibly equal to the `seed` argument) that was set for the random number generator before any sampling was undertaken. Note that the `i`th entry of this `seeds` attribute is the same as the `seed` attribute of the `i`th entry of the returned valuel

If `nrep==1` then the returned value is a single numeric scalar which is the log likelihood of the validation data or `NA` if the fitting procedure did not converge for the training data. It has an attribute `seed` which is equal to the `seed` argument or its random replacement. If the value is `NA` then it has a further attribute `lastPar`. (See above.)

Note

If the function fails to fit the model, obtained from the training data, to the validation data, then the value returned is `NA`. This value will have an attribute `lastPar`. This attribute will in turn have attributes, `trnDat` and `valDat`, the training data and validation data which were being used in the failed fitting procedure. Supplying an appropriate value of `lastPar` enables the continuation of the fitting procedure, starting from where the procedure previously left off. See **Details** for a little more information.

Author(s)

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

References

Celeux, Gilles and Durand, Jean-Baptiste (2008). Selecting hidden Markov model state number with cross-validated likelihood. *Computational Statistics* **23** 541–564, `DOI 10.1007/s00180-007-0097-1`.

Smyth, Padhraic (2000). Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and Computing* **9** 63–72.

See Also

`eglhmm()`

Examples

```
## Not run:
ids   <- paste0("s",1001:1101)
cc  <- ccSim[ccSim$id
cc$id <- factor(cc$id)
cvll1 <- vector("list",9)
set.seed(42)
SEEDS <- sample(1:1e6,9)
for(k in 1:9) {
    cat("k =",k,"started\n")
    fit  <- eglhmm(categMC ~ 1, distr="M", method="em", data=cc, K=k,
                   itmax=1500,cells="id",verb=TRUE)
    cvll1[[k]] <- crossval(fit,nrep=5,type=2,seed=SEEDS[k],tolerance=1e-4,
                           verbose=FALSE,crossVerb=TRUE)
    cat("k =",k,"finished\n")
}

## End(Not run)
fit   <- eglhmm(y ~ 1, data=Downloads,K=4,distr="P",verb=TRUE,cf="singlecell")
# Use artifically low value of itmax so that crossval() fails to
# fit the model to the training data
cvll2 <- crossval(fit,nrep=5,type=1,verbose=TRUE,seed=322596,itmax=5)
cvll3 <- crossval(fit,type=1,verbose=TRUE,lastPar=attr(cvll2[[1]],"lastPar"))
# So cvll3 carried on, in one instance, from where the first
# attempted fit in cvll2 gave up.
```

`eglhmm` *Fit (extended) generalised linear hidden Markov models.*

Description

Fits an (extended) generalised linear model to a data set where the response in each "cell" of the model consists of a time series whose serial dependence is modelled by a hidden Markov model.

Usage

```
eglhmm(formula = NULL, response = NULL, data,
      distr = c("Gaussian", "Poisson", "Binomial", "Dbd", "Multinom", "discnp"),
    inclTau=TRUE,preSpecSigma=NULL, indep = NULL, size = NULL, nbot = NULL, ntop = NULL,
      cells = NULL, cf = "singlecell", K = NULL, par0 = NULL, randStart = NULL,
      method = c("lm", "em", "bf"), optimiser = c("optim", "nlm"),
      optimMethod = "BFGS", nlmWarn = FALSE, lmc = 10, tolerance = NULL,
      digits = NULL, verbose = FALSE, itmax = 200,
      contrast = c("treatment", "sum", "helmert"),
      crit = c("CLL", "L2", "Linf", "ABSGRD"), breaks = NULL, hessian = FALSE,
      useAnalGrad = FALSE, ca = FALSE, checkDecrLL=TRUE)
```

Arguments

formula	A model formula specifying the linear predictor for the model. The formula should *not* include state as a predictor variable. The variable state gets added to the formula automatically. Ignored if the model is bivariate, i.e. if the length of response is 2.
response	A character scalar or a length-2 vector of such scalars, specifying the name or names of the response(s). If response is not specified (i.e. if it is left as NULL) then formula (see below) *must* be specfied and response is taken to be the left hand side of formula. (In this case, it is of course univariate.)
data	A data frame with columns providing the response(s) and the predictor variables in the model.
distr	Character string specifying the distribution of the response(s) ("emissions") variable(s). Currently (13/02/2024) the only distributions accommodated are Gaussian, Poisson, Binomial, Dbd, and Multinom. Note that "discnp" is just an alternative expression for "Multinom". Ignored if the response is bivariate, in which case distr is forcibly set equal to "Multinom". I.e. bivariate models are, currently, fitted only to data in which the emissions have the "Multinom" distribution.
inclTau	Logical scalar. Should the transition probability matrix parameters "tau" be included in those that are estimated via the Hessian/gradient pardigm? In this case, they are included in the set of parameters to which the gradient and Hessian are applicable. If not, they are estimated via the method of moments as is done when the EM algorithm is used. In this latter case the dimensions of the Hessian are reduced (by a substantial amount if K is "large").
preSpecSigma	Numeric vector of length K (see below) with strictly positive entries. Ignored if distr is not equal to "Gaussian". This vector provides "pre-specified" values of the standard deviations sigma of the Gaussian distribution associated with each state. If preSpecSigma is specified, then it is used as the value of sigma throughout the fitting process, and sigma is *not* estimated from the data. If distr is "Gaussian" and preSpecSigma is specified, then an error will be thrown if the length of preSpecSigma is not equal to K, or if any entries of preSpecSigma fail to be strictly positive.
indep	Logical scalar; should the components of a bivariate model be considered to be independent? Ignored unless the model is bivariate (i.e. unless response is of length 2. If the model is bivariate and indep is not specified, an error is thrown.
size	Scalar integer specifying the number of trials in the experiment generating binomial data (see the size argument of dbinom()). Ignored unless distr is equal to "Binomial".
nbot	Scalar integer specifying the lower end (0 or 1) of the range of values of the discretised Beta distribution. Ignored unless distr is "Dbd".
ntop	Scalar integer specifying the upper end of the range of values of the discretised Beta distribution. Ignored unless distr is "Dbd".
cells	A character vector giving the names of the factors (columns of the data data frame) which determine what the "cells" of the model are considered to be. The cells correspond to the combinations of levels of the factors named by cells.

The sequences of observations from each of the cells constitute a collection of independent time series, all following the specified model.

`cf` A factor ("cell factor") specifying the cells of the model. If `cells` is not specified, then `cf` must be. If `cells` *is* specified, then `cf` is ignored. the model. If `cells` is not specified, then in most (if not all?) circumstances, `cf` should be set equal `factor(rep(1,nrow(data))`. This the effect of making the entire observation sequence equal to a single time series, following the specified model.

`K` Scalar integer specifying the number of states of the hidden Markov model in question. If `K` is not specified and `par0` (see below) *is* specified, and has a component `tpm`, then `K` is set equal to `nrow(tpm)`. In this case, if `par0` does not have a `tpm` component, an error is thrown. An error is also thrown in this setting if `K` is specified to a value different from `nrow(tpm)`.

`par0` A list comprising starting values for the parameter estimates, to be used by the various methods. (See `method` below.) This list may have components `tpm` (an estimate of the transition probability matrix), `phi` (a vector of estimates of the coefficients in the linear predictor in the generalised linear model) and `Rho` (a matrix, a list of two matrices, or a three dimensional array) that specifyies the emission probabilities when `distr` is `"Multinomial"`. Note that `par0` may consist of an object of class `"eglhmm"` (see below), i.e. a model previously fitted (perhaps without achieving convergence), by `eglhmm()`. This provides a means whereby a fitting procedure, that failed to converge, may be continued from where it left off.

`randStart` Either a logical scalar or a list of three logical scalars named `tpm`, `phi`, and `Rho`. If the former, it is converted internally into a list with entries named `tpm`, `phi` and `Rho`, all having the same value as the original argument. If `tpm` is TRUE then the (undocumented) function `inititialise()` chooses entries for the starting value of `tpm` at random; likewise for `phi` and `Rho`. If left `NULL`, this argument defaults to `list(tpm=FALSE,phi=FALSE,Rho=FALSE)`.

`method` Character string specifying the method used to fit the model. This may be `"lm"` (Levenberg-Marquardt algorithm), `"em"` (EM algorithm) or `"bf"` ("brute force"). The latter calls upon `optim()` or `nlm()` to do the heavy lifting). If the response is bivariate, then `method` is forcibly (and silently) set equal to `"em"`.

`optimiser` Character string specifying which of `optim()` or `nlm()` should be used when `method` is `"bf"`. Ignored unless `method` is `"bf"`.

`optimMethod` Character string specifying the optimisation method to be used by `optim()`. See `optim()` for details. Ignored unless `method` is `"bf"` and `optimiser` is `"optim"`.

`nlmWarn` The `nlm()` function sometimes produces, in the first few iterations, warnings to the effect "NA/Inf replaced by maximum positive value". These warnings are almost surely irrelevant and are annoying. If `nlmWarn` is `FALSE` (the default) then these warnings are suppressed. This argument is provided to allow for the remote possibilty that the user might want to see these warnings.

`lmc` Positive numeric scalar. The initial "Levenberg-Marquardt constant". Ignored unless `method` is `"lm"`.

`tolerance` Positive numeric scalar. The convergence tolerance to be used. What this value actually *means* depends upon `method`. If left as `NULL` it defaults to `1e-6` for the bivariate methods, to `sqrt(.Machine$double.eps` for the `"em"` and `"lm"`

methods, and to the default value of reltol used by optim() when method is "bf" and optimiser is "optim". It is ignored if method is "bf" and optimiser is "nlm".

digits — Integer scalar. The number of digits to which "progress reports" are printed when verbose (see below) is TRUE. There is a "sensible" default which is calculated in terms of tolerance. This argument is ignored if method is "bf".

verbose — Logical scalar; if TRUE, rudimentary "progress reports" are printed out at appropriate points during the iteration process. The nature of these "reports" varies with method.

itmax — Integer scalar. The maximum number of iterative steps to take. Has a somewhat different meaning when method is "bf", in which case the meaning depends on optimiser. For methods "em" and "lm", if convergence is not achieved by itmax steps, the function gives up, prints a message to this effect, and returns a value with a component converged=FALSE. This returned value may be used as a starting (the value of the argument par0) so that the iterations may be continued from where they left off. Unfortunately this facility is not available when method is "bf".

contrast — Text string specifying the contrast (in respect of unordered factors) (see contrasts() and options()) that will be used when the design matrix is constructed from the model formula. May be abbreviated (e.g. to "t", "s" or "h").

crit — Text string specifying the stopping criterion to be used. Possible values are "CLL" (scaled change in log likelihood), "L2" (scaled square root of the sum of squares of the changes in the parameter estimates), "Linf" (scaled maximum of the absolute value of the changes in the parameter estimates), and "ABSGRD" (scaled maximum of the absolute values of the entries of the gradient vector). The latter only makes sense for the Levenberg-Marquardt algorithm.

This argument is ignored if method is "bf". It seems that the "bf" method effectively uses "CLL" when optimiser is "optim". When optimiser is "nlm" it seems that a combination of (something like) "ABSGRD" and "CLL" is used.

breaks — A vector of K+1 values used to construct a set of guesses at the states corresponding to each observation. These are in turn used to calculate an initial estimate of the transition probability matrix. There is a "sensible" default (produced by the undocumented function breaker().

hessian — Logical scalar; should a Hessian matrix obtained by numerical differentiation be returned? Ignored unless method is "bf".

useAnalGrad — Logical scalar; should "analytical" calculation of the gradient be conducted? This argument is ignored unless the method is "bf".

ca — Logical scalar; "check analyticals". Used only when the method is "bf" and optimiser is "nlm", and is passed on to nlm().

checkDecrLL — Logical scalar; "check for a *decrease* in the log likelihood". Ignored unless the method is "em". Should the software check for a decrease in the log likelihood after an EM step? See the **Remarks** for further discussion.

Value

An object of class "eglhmm", consisting of a list with components:

`call` The call by which this object was created. Present so that `update()` can be applied to objects returned by `eglhmm()`.

`tpm` The estimated transition probability matrix.

`ispd` The estimated initial state probability distribution.

`phi` Except for the `"Multinom"` distribution this is the vector of estimated coefficients of the linear predictor in the generalised linear model. For the `"Multinom"` distribution it consists of the entries of `Rho` (see below) with the final all-zero column remove. In this case `phi` is of course redundant.

`theta` The vector of parameter estimates that the estimation procedure actually works with. It consists of the catenation of the non-redundant parameterization of the transition probability matrix and the vector `phi`. It is redundant in the case of the `"Multinom"` distribution.

`Rho` A matrix, or a list of two matrices or a three dimensional array specifying the emissions probabilities for a multinomial distribution. Present only if `distr` is `"Multinom"`.

`log.like` The value of the log likelihood of the model evaluated at the parameter estimates, i.e. the (approximately) maximal value of the log likelihood.

`gradient` (Not present for the `"em"` method.) The gradient vector of the log likelihood at the final parameter estimates; it *should* be effectively the zero vector.

`numHess` (Present only if `method` is `"bf"` and only if the argument `hessian` is `TRUE`.) A value of the Hessian matrix (see below), obtained by means of numerical differentiation.

`Hessian` (Present only if `method` is `"lm"`.) The Hessian matrix, i.e. the matrix of second partial derivatives of the log likelihood, evaluated at the final parameter estimates. The inverse of the negative of this matrix constitutes an estimate of the covariance matrix of the parameter estimates.

`mu` A data frame with `npred+1` columns where `npred` is the number of predictors in the model. The rows contain, in their first `npred` entries, all possible combinations of the predictor values. The last (`npred+1`) entry of each row is the fitted mean of the Gaussian distribution, as determined by that combination of predictors. Present only if `distr` is `"Gaussian"`.

`sigma` Numeric vector of length `K` whose entries consist of the fitted standard deviations for the underlying Gaussian distribution, corresponding to each of the states. Present only if `distr` is `"Gaussian"` and `preSpecSigma` is *not* supplied.

`preSpecSigma` Numeric vector equal to the `preSpecSigma` argument, with names `"sigma1"`, `"sigma2"`, ..., `"sigmaK"` added. Present only if `distr` is `"Gaussian"` and `preSpecSigma` is supplied.

`stopCritVal` Numeric scalar equal to the value, assumed by the stopping criterion specified by the argument `crit`, at the termination of the algorithm. If the algorithm converged then `stopCritVal` will be less than `tolerance`. Not present if `method` is `"bf"`. If `converged` (see below) is `NA` then `stopCritVal` is `NA` also.

`anomaly` Logical scalar. Did an "anomaly" occur in an application of the EM algorithm? (See **Remarks**.) ' Present only if `method` was equal to `"em"`. This entry of the returned value is provided mainly for use by the `bcov()` function. Note that `anomaly` is added to the returned object, irrespective of the

value of checkDecrLL. When checkDecrLL is TRUE, anomaly is somewhat redundant, since it will be TRUE if aand only if converged is NA. However when checkDecrLL is FALSE, anomaly is informtive, since it is not possible to tell from other entries of the returned value when an anomaly has occurred.

converged — A logical scalar. For the "lm", and "em" methods it is TRUE if convergence is achieved within itmax iterations and FALSE otherwise. For the "em" method, if checkDecrLL is TRUE, then converged may be NA. See **Remarks** for some discussion.

For the "bf" method converged is TRUE if the convergence component of the object returned by optim() is equal to 0 or if the code component of the object returned by nlm() is less than or equal to 2, and is FALSE otherwise. When nlm() is used, the value of converged has an attribute "code" equal to the actual value of the code component.

nstep — The number of steps (iterations) actually used by the algorithm. For the "lm" and "em" methods this is the number of Levenberg-Marquardt steps, or EM steps, respectively, taken by the algorithm. For the "bf" method it is the counts component of the object returned by optim() when optimiser is "optim" and it is the iterations component of the object returned by nlm() when optimiser is "nlm".

mean — A vector of the fitted mean values underlying each combination of observed predictors and state (i.e. corresponding to each entry of y in the data frame used to fit the model. See the description of data below. Present only if distr is "Gaussian".

sd — A vector of the fitted values of the standard deviations underlying each combination of observed predictors and state, i.e. corresponding to each entry of y in the data frame used to fit the model. See the description of data below. Present only if distr is "Gaussian".

lambda — A vector of estimated values of the Poisson parameter associated with each combination of observed predictors and state, i.e. corresponding to each entry of y in the data frame used to fit the model. See the description of data below. Present only if distr is "Poisson".

p — A vector of estimated values of the "success" probabilities associated with each combination of observed predictors and state, i.e. corresponding to each entry of y in the data frame used to fit the model. See the description of data below. Present only if distr is "Binomial".

alpha — A numeric vector of the fitted "alpha" parameters, of the discretised Beta distribution, corresponding to each observation. Present only if distr is "Dbd".

beta — A numeric vector of the fitted "beta" parameters, of the discretised Beta distribution, corresponding to each observation. Present only if distr is "Dbd".

fy — The values of the "emission probability (density)" function, calculated at each observed value, for each state (i.e. at each entry of y in data. See below.) These values are calculated using the (final) fitted parameters.

message — A (long) text string that is produced if the EM algorithm encounters the anomaly of a decrease in the log likelihood after an EM step. It warns the user that this has occurred and suggests consulting the help file for an explanation. Present

only if method=="em", the anomaly referred to has occurred, and checkDecrLL is TRUE.

par0 — The starting values used in the estimation procedure. Either those provided by the argument par0 or those created by the (undocumented) function initialise.

cells — A character vector indicating the names of the factors specifying the "cells" of the model. (Equal to the cells argument.)

formula — The formula for the model that was fitted; equal to the formula argument, augmented by state.

distr — Text string specifying the distribution of the response variable. Equal to the distr argument of this function.

nbot — Integer scalar. The lower endpoint of the range of values of the discretised beta distribution. Equal to the value of the nbot argument of this function. Present only if distr is "Dbd".

ntop — Integer scalar. The upper endpoint of the range of values of the discretised beta distribution. Equal to the value of the nbot argument of this function. Present only if distr is "Dbd".

size — Scalar integer equal to the number of trials in the "experiments" generating the data. Equal to the size argument of this function. Present only if distr is "Binomial".

tolerance — The convergence tolerance used to fit the model. Equal to the tolerance argument.

crit — Character scalar specifying the stopping criterion that was used. Equal to the crit argument of this function. Not present if method is "bf".

contrast — Text string specifying the contrast for unordered factors that was used in fitting the model. Equal to the contrast argument of this function.

method — The method ("lm", "em", or "bf" used to fit the model. Equal to the method argument.

stationary — Logical scalar. Was a stationary Markov chain fitted? Currently (13/02/2024) stationary is always TRUE.

data — The data frame to which the model was fitted. It is a rearrangement of the data argument, with rows of that argument replicated K times (once for each state). A state column (factor) has been added, as has a column cf ("cell factor"), which indicates, by means of a single factor, which cell of the model a given row of data corresponds to. The aforementioned rearrangement consists of ordering the cells in the order of the levels of cf. When distr is "Multinom" the "response" variables are coerced into factors.

bicm — Numerical 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 eglhmm() for details.

AIC — Numerical scalar. The Akaike Information criterion, calculated as -2*ll + 2*npar where ll is the log likelihood of the fitted model and npar is the number of fitted parameters.

BIC — Numerical scalar. The Bayesian Information criterion, calculated as -2*ll + bicm*npar where ll is the log likelihood of the fitted model, npar is the number of fitted parameters, and bicm is the log of the number of observations.

missFrac — The fraction or proportion of missing values in the observations.

Remarks

Available models: Although this documentation refers to (extended) "generalised linear models", the only such models currently (13/02/2024) available are the Gaussian model with the identity link, the Poisson model, with the log link, and the Binomial model with the logit link. When `distr` is `"Dbd"` or `"Multinom"` the model fitted is is a generalised linear model only in a rather extended sense. Even the Gaussian model is not strictly speaking a generalised linear model, since the (state dependent) standard deviations are estimated by a method separate from the generalised linear model paradigm. Other models may be added at a future date.

Decrease in the log likelihood: 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 `method` is equal to `"em"`, then the object returned by `eglhmm()` has a component `anomaly` which is `TRUE` if such a decrease in the log likelihood was detected, and `FALSE` otherwise.

If such a decrease/anomaly is detected, then (provided that `checkDecrLL` is `TRUE`) the algorithm terminates and the `converged` component of the returned value is set equal to `NA`. The algorithm 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. This of course is not possible if the response is bivariate, in which case only the EM algorithm is applicable.

Note that if `checkDecrLL` is `FALSE`, then the algorithm proceeds "normally". That is, it treats the decrease in the log likelihood to mean that the "increase" in the log likeihood is less than `tolerance` and deems convergence to be achieved.

The value of `checkDecrLL` is set to `FALSE` in the function `bcov()` so as to speed up the rate at which the iterations proceed. In other circumstances it is probably judicious to leave it at its default value of `TRUE`.

Author(s)

Rolf Turner <rolfturner@posteo.net>

References

T. Rolf Turner, Murray A. Cameron, and Peter J. Thomson (1998). Hidden Markov chains in generalized linear models. *Canadian Journal of Statistics* **26**, pp. 107 – 125, DOI: https://doi.org/10.2307/3315677.

Rolf Turner (2008). Direct maximization of the likelihood of a hidden Markov model. *Computational Statistics and Data Analysis* **52**, pp. 4147 – 4160, DOI: https://doi.org/10.1016/j.csda.2008.01.029

See Also

`fitted.eglhmm()` `reglhmm.default()` `reglhmm.eglhmm()` `bcov()`

Examples

```
loc4 <- c("LngRf","BondiE","BondiOff","MlbrOff")
```

```
SCC4 <- SydColCount[SydColCount$locn %in% loc4,]
SCC4$locn <- factor(SCC4$locn) # Get rid of unused levels.
rownames(SCC4) <- 1:nrow(SCC4)
fitP.em <- eglhmm(y~locn+depth,data=SCC4,distr="P",cells=c("locn","depth"),
                K=2,method="em",verb=TRUE)
## Not run:
    fitP.lm <- eglhmm(y~locn+depth,data=SCC4,distr="P",cells=c("locn","depth"),
                    K=2,verb=TRUE)
    fitD.lm <- eglhmm(formula=y~ma.com+nh.com+bo.com,data=SCC4,nbot=0,ntop=11,
                   cells=c("locn","depth"),distr="Dbd",K=2,method="lm",verb=TRUE,
                   tolerance=NULL)
    SCD4 <- SydColDisc[SydColDisc$locn %in% loc4,]
    SCD4$locn <- factor(SCD4$locn) # Get rid of unused levels.
    fitM.lm  <- eglhmm(formula=y~ma.com+nh.com+bo.com,data=SCD4,
                   cells=c("locn","depth"),distr="Multinom",K=2,
                   verb=TRUE)
    xxx <- split(SCD4,f=SCD4$locn)
    X   <- with(xxx,data.frame(y.LngRf=LngRf$y,y.BondiE=BondiE$y,depth=LngRf$depth))
    fitBiv <- eglhmm(response=c("y.LngRf","y.BondiE"),data=X,K=2,cells="depth",
                    indep=FALSE,verb=TRUE)

## End(Not run)
# See the help for ionChannelData for more examples involving the
# ion channel data.
```

`fitted.eglhmm` *Predict method for extended generalised linear hidden Markov models.*

Description

Predicted values based on an extended generalised linear hidden Markov model object.

Usage

```
## S3 method for class 'eglhmm'
fitted(object, ...)
```

Arguments

`object`	An object of class `eglhmm` as returned by `eglhmm()`.
`...`	Not used.

Value

A vector of fitted values of the same length as that of the observed values (i.e. length equal to the row dimension of the data frame to which the model was fitted. This data frame is equal to `object$data` but with repeated rows corresponding to different states collapsed to a single row. The row dimension of this data frame is thus `nrow(object$data)/K` where `K` is the number of states in the model. This data frame, with columns `cf` and `state` omitted, is returned as an attribute `data` of the vector of fitted values.

Remark

Although this documentation refers to "generalised linear models", the only such models currently (13/02/2024) available are the Gaussian model with the identity link, the Poisson model, with the log link, and the Binomial model with the logit link. Other models may be added at a future date.

Author(s)

Rolf Turner <rolfturner@posteo.net>

References

See the help for eglhmm() for references.

See Also

reglhmm() reglhmm.default() reglhmm.eglhmm() bcov()

Examples

```
loc4 <- c("LngRf","BondiE","BondiOff","MlbrOff")
SCC4 <- SydColCount[SydColCount$locn %in% loc4,]
SCC4$locn <- factor(SCC4$locn) # Get rid of unused levels.
rownames(SCC4) <- 1:nrow(SCC4)
fit <- eglhmm(y~locn+depth,data=SCC4,cells=c("locn","depth"),
              K=2,distr="P",contr="sum",verb=TRUE)
fv  <- fitted(fit)
with(attr(fv,"data"),plot(y[locn=="BondiOff" & depth=="40"],
         xlab="time",ylab="count"))
with(attr(fv,"data"),lines(fv[locn=="BondiOff" & depth=="40"]))
```

---

hydroData *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).

Format

Data frames with observations on the following 5 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 sets are named `linLandFlows`, `ftLiardFlows`, `portMannFlows`, `portMannSedLoads` and `portMannSedCon`.

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 <- eglhmm(deciles ~ 1,K=4,distr="M",data=linLandFlows,
              method="em",itmax=10,verb=TRUE)
```

ionChannelData *Ion channel data*

Description

Time series of observations, made by means of patch clamps, of current in picoamps, across cell membranes.

Notes

The data sets are named `ic25kHz_12_sgmnt1`, `ic25kHz_13_sgmnt2`, `ic25kHz_14_sgmnt2`, `ic25kHz_15_sgmnt2`, `ic50kHz_06_sgmnt2`, `ic50kHz_08_sgmnt2`, `ic50kHz_09_sgmnt1` and `ic50kHz_10_sgmnt1`.

These data are **not** immediately available in the `eglhmm` package. Their presence would cause the size of the `data` directory to exceed 4.5 Mb., which is unacceptably large. Consequently these data sets have been placed in a separate "data only" package called `ionChannelData`, which is available from `github`. This package may be obtained by executing the command:

```
install.packages("ionChannelData",repos="https://rolfturner.r-universe.dev")
```

After having installed the `ionChannelData` package, you may load it via `library(ionChannelData)` and then access the data sets in the usual way, e.g. `X <- ic25kHz_12_sgmnt1`.

Alternatively (after having installed the `ionChannelData` package) you may use the `::` syntax to access a single data set, e.g. `X <- ionChannelData::ic25kHz_12_sgmnt1`.

You can access the documentation via, e.g. `?ionChannelData::ionChannelData`.

miscprints *Specialised print methods.*

Description

Print objects of class `"RhoExpForm"`, `"RhoProbForm"` and `"kitty"`, appropriately.

Usage

```
## S3 method for class 'RhoExpForm'
print(x, ...)
## S3 method for class 'RhoProbForm'
print(x, ...)
## S3 method for class 'kitty'
print(x, ...)
```

Arguments

`x`	An object of class `"RhoExpForm"`, `"RhoProbForm"` or `"kitty"` respectively.
`...`	Not used. Present for compatibility with the generic `print()` function.

Details

The methods `print.RhoExpForm()` and `print.RhoProbForm()` are present essentially for debugging purposes only. The method `print.kitty()` is present to improve the appearance of printed output from `eglhmm` when there is a `"message"` component of this output. None of these methods would normally be called by users.

Value

None.

Author(s)

Rolf Turner `<rolfturner@posteo.net>`

`monoCyteSim` *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 Northland District Health Board system. The real data must be kept confidential due to ethics constraints.

Notes

The data sets are named `bivarSim` and `cSim`.

These data are **not** immediately available in the `eglhmm` package. Their presence would cause the size of the `data` directory to exceed 4.5 Mb., which is unacceptably large. Consequently these data sets have been placed in a separate "data only" package called `monoCyteSim`, which is available from `github`. This package may be obtained by executing the command:

```
install.packages("monoCyteSim",repos="https://rolfturner.r-universe.dev")
```

After having installed the `monoCyteSim` package, you may load it via `library(monoCyteSim)` and then access the data sets in the usual way, e.g. `X <- ccSim`.

Alternatively (after having installed the `monoCyteSim` package) you may use the `::` syntax to access a single data set, e.g. `X <- monoCyteSim::ccSim`.

You can access the documentation via, e.g., `?monoCyteSim::ccSim`.

`plot.eglhmm` *Plot the results of an eglhmm fit.*

Description

For each specified model cell plot an array, with one panel for each state, of the probability mass or density functions corresponding to the given cell and state. The plots are produced with `type="h"` for probability mass functions, and with `type="l"` for probability density functions.

Usage

```
   ## S3 method for class 'eglhmm'
plot(x, ..., wcells = NULL, col = "red",
            nrnc = NULL, ntop = NULL, xlab = NULL, ylab = NULL,
            xlim = NULL, ylim = NULL, main = NULL, cex.main = 1.5)
```

Arguments

`x`	An object of class `"eglhmm"` as returned by the function `eglhmm()`.
`...`	Not used.
`wcells`	Character vector specifying the cells of the model to be plotted. Defaults to all cells (i.e. the levels of `x$data$cf`).
`col`	The colour for the (vertical) lines of the plots.
`nrnc`	An integer vector of length two specifying the dimenions of the array of plots that is produced. The first entry is the number of rows, the second the number of columns. The product of the entries must be greater than or equal to `K`, the number of states in the model.
`ntop`	The largest `x`-value to be used in plots of the Poisson distribution. Defaults to the maximum of the upper 10^{-7} quantile of all of the Poisson distributions that are to be plotted. Ignored unless `x$distr` is `"Poisson"`.
`xlab`	An optional label for the x-axes of the panels in the array of plots. Defaults to `"x"`.
`ylab`	An optional label for the y-axes of the panels in the array of plots. Defaults to `"probability"` for probability mass functions and to `"probability density"` for probability density functions.
`xlim`	An optional vector of length two, specifying the `x`-limits for the plots. Defaults to `c(0,ntop)` if `x$distr` is `"Poisson"` and to `c(0,x$size)` if `x$distr` is `"Binomial"`. There is no default if `x$distr` is `"Gaussian"`.
`ylim`	An optional vector of length two, specifying the `y`-limits for the plots. Defaults to `c(0,M)` where `M` is the maximum of all of the probabilities or probability density values that are to be plotted.
`main`	Optional character vector specifying overall titles for each array panel of plots. Defaults to the names of the model cells. If the length of `main` is less than the number (`nwc`) of cells to be plotted, then main is replicated to have length `nwc`.

If the length of main is greater than nwc then entries with index greater than nwc are ignored. If you wish there to be no overall titles for the arrays of plots, specify main="".

cex.main — Expansion factor for the text in the main title (determining the size of the text). Ignored if main is set equal to the empty string.

Details

If plotting is interactive, then the arrays of plots are displayed one at a time, and (except for the last of the plots) the user is prompted with the string "Go?" after each array is plotted. Press <return> to see the next plot.

Value

None.

Author(s)

Rolf Turner <rolfturner@posteo.net>

See Also

eglhmm()

Examples

```
loc4 <- c("LngRf","BondiE","BondiOff","MlbrOff")
SCC4 <- SydColCount[SydColCount$locn %in% loc4,]
SCC4$locn <- factor(SCC4$locn) # Get rid of unused levels.
rownames(SCC4) <- 1:nrow(SCC4)
fit <- eglhmm(y~locn+depth,data=SCC4,cells=c("locn","depth"),
            K=2,distr="P",verb=TRUE)
plot(fit)
allcells <- levels(fit$data$cf)
wcells   <- allcells[grep("\\.60",allcells)]
plot(fit,wcells=wcells,main=c("Longreef","Bondi East","Bondi Offshore",
                              "Malabar Offshore"),ntop=12)
```

---

postHocGradHess *Obtain gradient and Hessian, post hoc.*

Description

Calculates the gradient and Hessian of the log likelihood of an extended generalised hidden Markov model, from the components of a "eglhmm" object, or in certain circumstances, simply extracts these quantities from that object.

Usage

```
postHocGradHess(object,inclTau=TRUE)
```

Arguments

object	An object of class "eglhmm" as returned by eglhmm().
inclTau	Logical scalar; should the vector of "tau" parameters be included in the parameters under consideration?

Details

If object is the result of fitting a bivariate model (i.e. if it inherits from "eglhmm.bivariate" then an error is thrown. (No gradient or Hessian is available in this case.)

If object$method is "lm" and if inclTau matches the value used in fitting method, then the appropriate gradient and Hessian have already been calculated and are simply extracted from object. If inclTau does not match the value used in fitting method, then the gradient and Hessian are recalculated (by the undocumented function getHgl()) with the value of inclTau being that specified by the function argument.

If object$method is "em" or "bf", then the gradient and Hessian are calculated (by the undocumented function getHgl()) with the value of inclTau being that specified by the function argument.

If object$method is "bf" then the gradient has been calculated numerically and the Hessian *may* have been calculated numerically (f the argument hessian of eglhmm() was set equal to TRUE). The corresponding value of the gradient will comprise a component, named "numGrad", of the list returned by this function. The corresponding value of the Hessian, if this was indeed calculated, will comprise a component, named "numHess", of the list returned by this function.

Value

A list with components

- gradient: The gradient of the log likelihood.
- Hessian: The Hessian of the log likelihood.
- numGrad: The numerically calculated gradient of the log likelihood. Present only if object$method is "bf".
- numHess: The numerically calculated Hessian of the log likelihood. Present only if object$method is "bf" and if argument hessian was set equal to TRUE in the call to eglhmm() that produced object.

Author(s)

Rolf Turner <rolfturner@posteo.net>

References

R. Nazim Khan, (2002). *Statistical modelling and analysis of ion channel data based on hidden Markov models and the EM algorithm*. Ph.D. thesis, the University of Western Australia, Crawley, WA 6009.

David Oakes, Direct calculation of the information matrix via the EM algorithm (1999). *Journal of the Royal Statistical Society*, series B, **61**, pp. 479 – 482.

Theodore C. Lystig and James P. Hughes (2002). Exact computation of the observed information matrix for hidden Markov models. *Journal of Computational and Graphical Statistics*, **11** (3), pp. 678 – 689.

Olivier Cappé and Eric Moulines (July 2005). Recursive computation of the score and observed information matrix in hidden Markov models, *IEEE Workshop on Statistical Signal Processing*, Bordeaux.

See Also

`eglhmm()`

Examples

```
fit.em <- eglhmm(y~locn+depth,data=SydColCount,distr="P",
                 cells=c("locn","depth"),K=2,method="em",verb=TRUE)
gh.em  <- postHocGradHess(fit.em) # Calculates using inclTau=TRUE.
## Not run:
    gh.em.noTau <- postHocGradHess(fit.em,inclTau=FALSE)
    fit.lm <- eglhmm(y~locn+depth,data=SydColCount,distr="P",
                     cells=c("locn","depth"),K=2,verb=TRUE)
    gh.lm  <- postHocGradHess(fit.lm) # Just extracts the relevant components.
    gh.lm.noTau  <- postHocGradHess(fit.lm,inclTau=FALSE)
    fit.bf <- eglhmm(y~locn+depth,data=SydColCount,distr="P",
                     cells=c("locn","depth"),K=2,method="bf",verb=TRUE,
                     hessian=TRUE)
    gh.bf  <- postHocGradHess(fit.bf) # Calculates using inclTau=TRUE; also
                                      # extracts numerically computed quantities.
    gh.bf.noTau <- postHocGradHess(fit.bf,inclTau=FALSE) # Calculates; also
                                                         # extracts numerically
                                                         # computed quantities.

## End(Not run)
```

`reglhmm` *Simulate data from a hidden generalised linear Markov model.*

Description

Takes a specification of the model and simulates the data from that model. The model may be specified in terms of the individual components of that model (the default method). The components include a data frame that provides the predictor variables, and various parameters of the model. For the `"eglhmm"` method the model is specified as a fitted model, an object of class `"eglhmm"`.

Usage

```
reglhmm(x,...)
## Default S3 method:
reglhmm(x, formula, response, cells=NULL, data=NULL, nobs=NULL,
                     distr=c("Gaussian","Poisson","Binomial","Dbd","Multinom"),
                        phi, Rho, sigma, size, ispd=NULL, ntop=NULL, zeta=NULL,
                         missFrac = 0, fep=NULL,
                         contrast=c("treatment","sum","helmert"),...)
## S3 method for class 'eglhmm'
reglhmm(x, missFrac = NULL, ...)
```

Arguments

x	For the default method, the transition probability matrix of the hidden Markov chain. For the "eglhmm" method, an object of class "eglhmm" as returned by the function eglhmm().
formula	The formula specifying the generalised linear model from which data are to be simulated. Note that the predictor variables in this formula must include a factor state, which specifies the state of the hidden Markov chain. Note also that this formula must determine a design matrix having a number of columns equal to the length of the vector phi of model coefficients provided in object (and to the length of psi in the case of the Gaussian distribution). If this condition is not satisfied, an error is thrown. It is advisable to use a formula specified in the manner y~0+state+... where ... represents the predictors in the model other than state. Of course phi must be supplied in a manner that is consistent with this structure.
response	A character vector of length 2, specifying the names of the responses. Ignored unless distr is "Multinom". If distr is "Multinom" and if response is provided appropriately, then the simulated data are bivariate multinomial.
cells	A character vector specifying the names of the factors which determine the "cells" of the model. These factors must be columns of the data frame data. (See below.) Each cell corresponds to a time series of (simulated) observations. If cells is not supplied (left equal to NULL) then the model is taken to have a single cell, i.e. data from a "simple" hidden Markov model is generated. The parameters of that model may be time-varying, and still depend on the predictors specified by formula.
data	A data frame containing the predictor variables referred to by formula, i.e. the predictors for the model from which data are to be simulated. If data is not specified, the nobs (see below) must be. If data is not specified then formula must have the structure y ~ state or preferably y ~ 0 + state. Of course phi must be specified in a consistent manner.
nobs	Integer scalar. The number of observations to be generated in the setting in which the generalised linear model in question is vacuous. Ignored if data is supplied.
distr	Character string specifying the distribution of the "emissions" from the model, i.e., of the observations. This distribution determines "emission probabilities".

`phi` A numeric vector specifying the coefficients of the linear predictor of the generalised linear model. The length of `phi` must be equal to the number of columns of the design matrix determined by `formula` and `data`. The entries of `phi` must match up appropriately with the columns of the design matrix.

`Rho` A matrix, or a list of two matrices or a three dimensional array specifying the emissions probabilities for a multinomial distribution. Ignored unless `distr` is `"Multinomial"`.

`sigma` A numeric vector of length equal to the number of states. Its ith entry is the standard deviation of the (Gaussian) distribution corresponding to the ith state. Ignored unless `distr` is `"Gaussian"`.

`size` Integer scalar. The number of trials (sample size) from which the number of "successes" are counted, in the context of the binomial distribution. (I.e. the `size` parameter of `rbinom()`.) Ignored unless `distr` is `"Binomial"`.

`ispd` An optional numeric vector specifying the initial state probability distribution of the model. If `ispd` is not provided then it is taken to be the stationary/steady state distribution determined by the transition probability matrix `x`. If specified, `ispd` must be a *probability* vector of length equal to the number of rows (equivalently the number of columns) of `x`.

`ntop` Integer scalar, strictly greater than 1. The maximum possible value of the db distribution. See `db()`. Used only if `distr` is `"Dbd"`.

`zeta` Logical scalar. Should zero origin indexing be used? I.e. should the range of values of the db distribution be taken to be `{0,1,2,...,ntop}` rather than `{1,2,...,ntop}`? Used only if `distr` is `"Dbd"`.

`missFrac` A non-negative scalar, less than 1. Data will be randomly set equal to `NA` with probability `miss.frac`. Note that for the `"eglhmm"` method, if `"miss.frac"` is not supplied then it is extracted from `object`

`fep` A list of length 1 or 2. The first entry of this list is a logical scalar. If this is `TRUE`, then the first entry of the simulated emissions (or at least one entry of the first pair of simulated emissions) is forced to be "present", i.e. non-missing. The second entry of `fep`, if present, is a numeric scalar, between 0 and 1 (i.e. a probability). It is equal to the probability that both entries of the first pair of emissions are present. It is ignored if the emissions are univariate. If the emissions are bivariate but the second entry of `fep` is not provided, then this second entry defaults to the "overall" probability that both entries of a pair of emission are present, given that at least on is present. This probability is calculated from `nafrac`.

`contrast` A character string, one of "treatment", "helmert" or "sum", specifying what contrast (for unordered factors) to use in constructing the design matrix. (The contrast for ordered factors, which is has no relevance in this context, is left at it default value of `"contr.poly"`.) Note that the meaning of the coefficient vector `phi` depends on the contrast specified, so make sure that the contrast is the same as what you had in mind when you specified phi!!! Note that for the `"eglhmm"` method, `contrast` is extracted from `x`.

`...` Not used.

Value

A data frame with the same columns as those of `data` and an added column, whose name is determined from `formula`, containing the simulated *response*

Remark

Although this documentation refers to "generalised linear models", the only such models currently (13/02/2024) available are the Gaussian model with the identity link, the Poisson model, with the log link, and the Binomial model with the logit link. The Multinomial model, which is also available, is not exactly a generalised linear model; it might be thought of as an "extended" generalised linear model. Other models may be added at a future date.

Author(s)

Rolf Turner <rolfturner@posteo.net>

References

T. Rolf Turner, Murray A. Cameron, and Peter J. Thomson (1998). Hidden Markov chains in generalized linear models. *Canadian Journal of Statistics* **26**, pp. 107 – 125, DOI: https://doi.org/10.2307/3315677.

Rolf Turner (2008). Direct maximization of the likelihood of a hidden Markov model. *Computational Statistics and Data Analysis* **52**, pp. 4147 – 4160, DOI: https://doi.org/10.1016/j.csda.2008.01.029

See Also

`fitted.eglhmm()` `bcov()`

Examples

```
    loc4 <- c("LngRf","BondiE","BondiOff","MlbrOff")
    SCC4 <- SydColCount[SydColCount$locn %in% loc4,]
    SCC4$locn <- factor(SCC4$locn) # Get rid of unused levels.
    rownames(SCC4) <- 1:nrow(SCC4)
    Tpm   <- matrix(c(0.91,0.09,0.36,0.64),byrow=TRUE,ncol=2)
    Phi   <- c(0,log(5),-0.34,0.03,-0.32,0.14,-0.05,-0.14)
    # The "state effects" are 1 and 5.
    Dat   <- SCC4[,1:3]
    fmla  <- y~0+state+locn+depth
    cells <- c("locn","depth")
# The default method.
    X     <- reglhmm(Tpm,formula=fmla,cells=cells,data=Dat,distr="P",phi=Phi,
                     miss.frac=0.75,contrast="sum")
# The "eglhmm" method.
    fit <- eglhmm(y~locn+depth,data=SCC4,cells=cells,K=2,
                  verb=TRUE,distr="P")
    Y   <- reglhmm(fit)
# Vacuous generalised linear model.
    Z   <- reglhmm(Tpm,formula=y~0+state,nobs=300,distr="P",phi=log(c(2,7)))
    # The "state effects" are 2 and 7.
```

`reorder.eglhmm` *Reorder the states of a fitted eglhmm model*

Description

Reorder the states of a fitted eglhmm model so that the state effects are in decreasing order.

Usage

```
## S3 method for class 'eglhmm'
reorder(x, ...)
```

Arguments

`x`	An object of class `"eglhmm"` as returned by the function `eglhmm()`.
`...`	Not used.

Details

The states of a fitted hidden Markov model are usually in a rather arbitrary order, which can sometimes make it difficult to compare different fits. This function reorders the states so that the state corresponding to the "largest state effect" comes first (and so on down the line). What is meant by "largest state effect" depends on whether the distribution used in the model is "Dbd". If the distribution is *not* "Dbd", then what is meant is simply the largest of those entries of `phi` which correspond to `state`. (The vector `phi` is the vector of coefficients of the linear predictor in the model. Note that, since the formula for the model is constructed as `y~0+state+...`, the "state" coefficients are unconstrained and there are as many of them as there are states.)

If the distribution in question *is* "Dbd" then things are a bit more complicated. We calculate the theoretical expected values for "Dbd"s with parameters $\alpha =$ `alpha[k]` and $\beta =$ `beta[k]` where `alpha[k]` and `beta[k]` are the parameter values corresponding to the `k`th state. The states are then ordered according to the decreasing order of these expected values. These expected values are the expected values of the emissions given that all predictors other than the *state* predictors are zero.

Value

An object of class `c("eglhmm","reordered")` which is identical to the argument `x` in most respects. The components which (may) differ are:

- `tpm`
- `ispd`
- `phi`
- `theta`
- `Hessian`
- `gradient`
- `mean` and `sd`, or `lambda` or `p`, or `alpha` and `beta` (depending on which distribution is being used)

- fy

The entries of these components will have the same numerical values as before but, given that the ordering of the states has actually changed, will have different orderings, corresponding to the new ordering of the states.

Note that the *attribute* preSpecSigma of the component theta, may differ from what it was in x.

The returned value will also have a component neworder which is an integer vector providing the indices of the reordering of the states. It also currently (13/02/2024) has a component newlog.like. This should (if there is any justice in the world — but there isn't!) have the same value as the component log.like. Once I am confident that everything is working as it should, the newlog.like component will be removed.

Author(s)

Rolf Turner <rolfturner@posteo.net>

See Also

eglhmm()

Examples

```
loc4 <- c("LngRf","BondiE","BondiOff","MlbrOff")
SCC4 <- SydColCount[SydColCount$locn %in% loc4,]
SCC4$locn <- factor(SCC4$locn) # Get rid of unused levels.
rownames(SCC4) <- 1:nrow(SCC4)
fit  <- eglhmm(y~locn+depth,data=SCC4,cells=c("locn","depth"),
              K=2,distr="P",verb=TRUE)
ofit <- reorder(fit)
```

SydColDat *Sydney coliform bacteria data*

Description

Transformed counts of faecal coliform bacteria in sea water at seven locations: Longreef, Bondi East, Port Hacking "50", and Port Hacking "100" (controls) and Bondi Offshore, Malabar Offshore and North Head Offshore (outfalls). At each location measurements were made at four depths: 0, 20, 40, and 60 meters.

The data sets are named SydColCount and SydColDisc.

Format

Data frames with 5432 observations on the following 6 variables.

`y` Transformed measures of the number of faecal coliform count bacteria in a sea-water sample of some specified volume. The original measures were obtained by a repeated dilution process.

For `SydColCount` the transformation used was essentially a square root transformation, resulting values greater than 150 being set to `NA`. The results are putatively compatible with a Poisson model for the emission probabilities.

For `SydColDisc` 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")`.

Note that in the `SydColDisc` data there are 180 fewer missing values (`NA`s) in the `y` column than in the `SydColCount` data. This is because in forming the `SydColCount` data (transforming the original data to a putative Poisson distribution) values that were greater than 150 were set equal to `NA`, and there were 180 such values.

`locn` a factor with levels "LngRf" (Longreef), "BondiE" (Bondi East), "PH50" (Port Hacking 50), "PH100" (Port Hacking 100), "BondiOff" (Bondi Offshore), "MlbrOff" (Malabar Offshore) and "NthHdOff" (North Head Offshore)

`depth` a factor with levels "0" (0 metres), "20" (20 metres), "40" (40 metres) and "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 time series. The sampling interval is ostensibly 1 week; distinct time series are ostensibly synchronous. The measurements were made over a 194 week period. See Turner et al. (1998) for more detail.

Source

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., vol. 26, pp. 107 – 125, 1998.

Rolf Turner. Direct maximization of the likelihood of a hidden Markov model. *Computational Statistics and Data Analysis* **52**, pp. 4147 – 4160, 2008, doi:10.1016/j.csda.2008.01.029.

Examples

```
# Select out a subset of four locations:
loc4 <- c("LngRf","BondiE","BondiOff","MlbrOff")
SCC4 <- SydColCount[SydColCount$locn %in% loc4,]
SCC4$locn <- factor(SCC4$locn) # Get rid of unused levels.
rownames(SCC4) <- 1:nrow(SCC4)
```

`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.

The data sets are named `Bovine`, `Cryptosporidiosis`, `Downloads`, `EricssonB_Jul2`, `FattyLiver`, `FattyLiver2`, `goldparticle380`, `Hanta`, `InfantEEGsleepstates`, `IPs`, `LegionnairesDisease`, `OffshoreRigcountsAlaska`, `PriceStability`, `Strikes` and `WoodPeweeSong`.

Format

Each data set is a data frame with a single column named `"y"`.

- `Bovine` There are 8419 rows. The column `"y"` is a factor, with levels `"a"`,`"c"`,`"g"`,`"t"`, the DNA "bases". It constitutes the DNA sequence of the bovine leukemia virus.
- `Cryptosporidiosis` There are 365 rows. The column `"y"` is a numeric (integer) vector. It consists of weekly counts of new infections, in Germany in the years 2002 to 2008. The counts vary between 2 and 78.
- `Downloads` There are 267 rows. The column `"y"` is a numeric (integer) vector. It consists of the daily number of downloads of a TEX editor for the period from June 2006 to February 2007. These counts vary between 0 and 14.
- `EricssonB_Jul2` There are 460 rows. The column `"y"` is a numeric (integer) vector. It consists of the number of transactions per minute, of the Ericsson B stock, between 9:35 and 17:14 on 2 July, 2002. The counts vary between 0 and 37.
- `FattyLiver` There are 928 rows. The column `"y"` is a numeric (binary) vector. The value 1 indicates that "the considered diagnosis cannot be excluded for the current patient; that is, suitable countermeasures are required", and the value 0 indicates that this is not so. The values refer to different patients, examined sequentially over time.
- `FattyLiver2` There are 449 rows. The column `"y"` is a numeric (binary) vector as for `FattyLiver`. (Different examiner, different sequence of patients.)
- `goldparticle380` There are 380 rows. The column `"y"` is a numeric (integer) vector of counts of gold particles measured in a fixed volume element of a colloidal solution over time. The count values vary because of the Brownian motion of the particles. They vary between 0 and 7.
- `Hanta` There are 52 rows. The column `"y"` is a numeric (integer) vector consisting of the weekly number of territorial units (out of $n = 38$ territorial units with at least one new case of a hantavirus infections, in the year 2011. The numbers vary between 0 and 11.
- `InfantEEGsleepstates` There are 107 rows. The column `"y"` is a factor with levels `qt`, `qh`, `tr`, `al`, `ah`, `aw`. The level `"aw"` does not actually appear.
- `IPs` There are 241 rows. The column `"y"` is a numeric (integer) vector of the counts of different IP addresses registered at a web server within periods of length two minutes, "assumed" to have been observed between 10:00 a.m. and 6:00 p.m. on 29 November 2005. The counts vary between 0 and 8.

- `LegionnairesDisease` There are 365 rows. The column `"y"` is a numeric (integer) vector of weekly counts of new infections in Germany, in the years 2002 to 2008. The counts vary between 0 and 26.
- `OffshoreRigcountsAlaska` There are 417 rows. The column `"y"` is a numeric (integer) vector of weekly counts of active rotary drilling rigs in Alaska for the period 1990 to 1997. The counts vary between 0 and 6.
- `PriceStability` There are 152 rows. The column `"y"` is a numeric (integer) vector of monthly counts of countries (out of a group of 17 countries) that showed stable prices (that is, an inflation rate below 2%), in the period from January 2000 to December 2006. The counts vary between 0 and 17.
- `Strikes` There are 108 rows. The column `"y"` is a numeric (integer) vector of the monthly counts of work stoppages (strikes and lock-outs) of 1000 or more workers in the period 1994 to 2002. The counts vary between 0 and 14.
- `WoodPeweeSong` There are 1327 rows. The column `"y"` is a factor with levels `"1"`, `"2"`, `"3"` corresponding to the three different "phrases" of wood wewee song. The time series comprises a sequence of observations of the "morning twilight" song of the wood pewee.

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