Package ‘RMC’

February 19, 2015

Title Functions for fitting Markov models
Version 0.2
Author Scott D. Foster
Description Functions for fitting, diagnosing and predicting from a
class of Markov models.
Maintainer Scott Foster <scott.foster@csiro.au>
License GPL (>= 2)
Repository CRAN
Date/Publication 2010-11-08 07:27:02
NeedsCompilation yes

R topics documented:

dataEG1 .......................................................... 2
dataEG2 .......................................................... 2
dataEG3movement ............................................. 2
dataEG3patch .................................................. 3
dataEG4 .......................................................... 3
diagnos ......................................................... 4
eexamplesForDiagnostics .................................... 6
hrplot ............................................................ 8
MVfill ............................................................ 10
RMC ............................................................. 11
RMC.mod ....................................................... 11
RMC.pred ..................................................... 14
sim.chain ..................................................... 16
simRandWalk .................................................. 18

Index  20
Simulated stationary data used in Foster and Bravington (2009) Section 5.1.1.

Description

These data are stationary and have 5 chains, each with 1000 observations. The first column of this data matrix is an indicator to identify the chains. The second column is the observations. The third and final column is a constant design matrix. The data were originally used to highlight the diagnostic methods, see `diagnos.envel`, `diagnos`, and `hrplot` for functions to perform those original tasks.

References


Simulated non-stationary data used in Foster and Bravington (2009) Section 5.1.1.

Description

These data are non-stationary and have 5 chains, each with 1000 observations. The first column of this data matrix is an indicator to identify the chains. The second column is the observations. The third and fourth columns are the design matrix for the process. The data were originally used to highlight the diagnostic methods, see `diagnos.envel`, `diagnos`, and `hrplot` for functions to perform those original tasks.

References


Simulated data, illustrating an outlying movement residual, used in Foster and Bravington (2009) Section 5.1.2.

Description

These data are stationary and have 5 chains, each with 1000 observations. The first column of this data matrix is an indicator to identify the chains. The second column is the observations. The third and final column is a constant design matrix. The outlying movement is 3 -> 4. The data were originally used to highlight the diagnostic methods, see `diagnos.envel`, `diagnos`, and `hrplot` for functions to perform those original tasks.
**References**


---

**dataEG3patch**

Simulated data, illustrating an outlying patch residual, used in Foster and Bravington (2009) Section 5.1.2.

---

**Description**

These data are stationary and have 5 chains, each with 1000 observations. The first column of this data matrix is an indicator to identify the chains. The second column is the observations. The third and final column is a constant design matrix. The outlying patch is of state 3 and occurs at locations 301:320 of the first chain. The data were originally used to highlight the diagnostic methods, see `diagnos.envel`, `diagnos`, and `hrplot` for functions to perform those original tasks.

**References**


---

**dataEG4**

Simulated non-stationary data used in Foster and Bravington (2009) Section 5.1.3.

---

**Description**

These data are non-stationary and have 5 chains, each with 1000 observations. The first column of this data matrix is an indicator to identify the chains. The second column is the observations. The third and fourth columns are the design matrix for the process.

These data are also used to illustrate the effect of missing covariates.

**References**

diagnos

Calculation of Markov residuals for discrete-time non-stationary Markov models with simple parameterisation.

Description

Calculates patch and movement residuals for Markov models with a simple parameterisation. The models themselves are for categorical Markov processes that are usefully described by models whose parameterisation is based on a simple reversible Markov model and that can be extended to non-stationary cases. Non-stationary models are incorporated by letting the transition matrix vary with covariates. Simulation envelopes are created using diagnos.envel.

Usage

diagnos(obsNstates, chainNid, X=NULL, fit)
diagnos.envel( obsNstates, chain.id, X=NULL, fit, perc=c(0.025, 0.975), B=100, contr=list(print.iter=1))

Arguments

obsNstates observed ordered chained data. If there are multiple chains then chains are stacked on top of each other. Argument must be supplied
chain.id vector (length matches states) of identifiers for the individual chains. If NULL then it is assumed that all observations form a single chain.
X design matrix as passed to the model fitting routine RMC.mod. If NULL then a matrix with 1 column full of ones is assumed.
fit a fitted model formed by a call to the estimation function RMC.mod. Must match up with the X argument.
perc the percentiles of the simulations to take for the simulation envelope
B the number of simulations for the simulation envelope
contr list describing control parameters for the function. Currently consists of a single value for how often printing is to be performed

Details

Calculates the patch and movement residuals for the given Markov model. The methods are given in Foster and Bravington (2009). The patch residuals assess the model’s compatibility with the data by inspecting the probabilities of observing each fully observed patch. The movement residuals assess the model’s ability to describe each of the movement categories in the transition matrix.

Usage of diagnos.envel will produce, in addition to the functionality of diagnos, simulation envelopes
**Value**

*for diagnos* A list with the following elements

- **patch**: a list with number of elements equal to the number of states. Each element contains the patch residuals for each state
- **movement**: a square matrix containing the movement residuals from and to each state
- **njumps**: a square matrix containing the number of jumps from each state to each other state

*for diagnos.envel* A list with the following elements

- **patch**: a list with elements equal to the number of states. Each list element contains a matrix with observed patch residuals, expected patch quantiles and, lower and upper simulation envelope bounds. All values are given on uniform and normal deviate scales
- **movement**: a matrix with observed movement residuals, expected movement residuals and, lower and upper simulation envelope bounds. All values are given on uniform and normal deviate scales
- **njumps**: a list with length equal to the number of states. Each element contains the number of observed and simulated movements from each a particular state to each other state

**Author(s)**

Scott D. Foster

**References**


**See Also**

- `rmcNmod` for estimation of the Markov models and `hrplot` to plot the calculated residuals from `diagnos.envel`.

**Examples**

```r
#estimate a model for stationary example data, dataEGI
fm.est <- rmc.mod( states=dataEGI[,2], chain.id=dataEGI[,1], X=dataEGI[,3])
#calculate residuals
res <- diagnos( dataEGI[,2], dataEGI[,1], X=dataEGI[,3, drop=FALSE], fit=fm.est)
res.envel <- diagnos.envel( dataEGI[,2], dataEGI[,1], X=dataEGI[,3,drop=FALSE], fit=fm.est, B=25)
```
The examples section of this help file provides code to produce the plots in Section 5.1 of Foster and Bravington (2009).

Description

This help file provides code to generate illustrative, simulated examples for different types of data/model discrepancies. The examples are those given in Section 5.1 of Foster and Bravington (2009). The code is provided so that any interested reader of that paper can, if they want, reproduce the results.

Author(s)

Scott D. Foster

References


See Also

*RMC.mod* for estimation of the Markov model, *diagnos* and *diagnos.envel* for calculation of residuals and simulation envelopes, *hrplot* for plotting the residuals, and *sim.chain* for simulating chained data.

Examples

```r
## common parameters for the examples
nc <- 5
n <- 1000
n.cats <- 4
B <- 50 #number of simulations for simulated envelopes. The paper uses 1000 but this can be pretty slow

### Example of stationary well-fitting models ###
set.seed(21)
#simulating data -- will be the same as dataEG1
chain <- sim.chain( n.chains=nc, n.obs=rep( n, nc), n.cats=n.cats, n.covars=1, beta=c(0,0.3,-0.3,0), gamma=c(0.5)
#plotting start of first chain as an example (Figure 1 of paper)
m <- 100
plot(1:m, head( chain[,"chain"]==2,"state"),m, type='b', pch=19, main="Start of Example Chain", ylab="State",
abline(h=c(1:n.cats), lty=3, col=grey(0.5))
axis(1)
axis( 2, 1:n.cats, 1:n.cats)
box()
#fitting the model
fm.est <- RMC.mod( states=chain[,2], chain.id=chain[,1], X=chain[,3])
```
# defining true model
fm <- fm.est
fm$par <- c(0.5,0.2,1,0.03,-0.3,0)

# generating simulation envelope
temp.est <- diagnos.envel( obs.states=chain[,2], chain.id=chain[,1], X=chain[,3], drop=FALSE, fit=fm.est, B=8)

# plotting patch residuals (Figure 2 of paper)
par(mfrow=c(1,2))
my.cat <- 2
hrplot(temp.est[["patch"]][[my.cat]], ylab="Patch Residuals - Normal Quantiles", xlab="Normal Quantiles", main="patch", my.cat <- 3
hrplot(temp.est[["patch"]][[my.cat]], ylab="Patch Residuals - Normal Quantiles", xlab="Normal Quantiles", main="patch"

# generating movement residuals (Figure 3 of paper)
hrplot(temp.est$movement, ylab="Movement Residuals - Normal Quantiles", xlab="Normal Quantiles", main="Movement"

## Example of non-stationary good well-fitting models ##

set.seed(10)

# simulating data -- will be the same as dataEG2
X <- cbind(rep(1, nc*nL), simRandWalk(nc=nc, ni=rep(n, nc), init.var=1, seq.var=0.1)[,-1])
colnames(X) <- c("const", "rand2")
n.covs <- 2
gpar <- matrix(c(c(-0.6, 0), (1.1, 1.5), (0.2, 0), (1.25, -1.5)), nrow=n.covs, ncol=n.cats)
bpar <- matrix(c(c(0, n.covs), (0.9, 0.5), (0.8, -0.4), (0.4, -0.7)), nrow=n.covs, ncol=n.cats)
chain.ns <- sim.chain(n.chains=nc, n.obs=rep(n, nc), n.cats=n.cats, n_covars=n.covs, beta=bpar, gamma=gpar, X=X)

# setting up model
my.phi.id <- ifelse(gpar!=0, 1, 0)
my.pi.id <- apply(bpar, FUN=function(x){if( any(x!=0)) 1 else 0}, MARG=1)

# fitting the model
fm.est1 <- RMC.mod( states=chain.ns[,2], chain.id=chain.ns[,1], X=chain.ns[,3:4], phi.id=my.phi.id, pi.id=my.pi.id)

# generating simulation envelope
temp1 <- diagnos.envel( chain.id=chain.ns[,1], obs.states=chain.ns[,2], X=chain.ns[,3:4], fit=fm.est1, B=B)

# plotting residuals (Figure 4 of paper)
par(mfrow=c(1,1))

# adding outliers ####

set.seed(25)

# simulating data -- will be the same as dataEG3patch and dataEG3movement
chain.orig <- sim.chain(n.chains=nc, n.obs=rep(n, nc), n.cats=n.cats, n.covars=1, beta=c(0.0.3,-0.3,0), gamma=c(0.0.3,-0.3))

chain1[301:320, "state"] <- 3
ids <- sample( seq( from = n, to = n*nc, by = n ) , 100)

chain2[ ids, "state"] <- 3

chain2[ ids+1, "state"] <- 4

# fit the models
fm1 <- RMC.mod( states=chain1[,2], chain.id=chain1[,1], X=chain1[,3])
fm2 <- RMC.mod( states=chain2[,2], chain.id=chain2[,1], X=chain2[,3])

# generate simulation envelopes
hrplot

Plot horizontal residual plots for a RMC model.

Description

Plot movement or a particular state’s patch residuals for objects created by the estimation function RMC.mod.

Usage

hrplot( envel.obj, ylim, ...)
Arguments

envel.obj a matrix of residuals and their upper and lower simulation limits. The easiest method to obtain this object is via a call to the diagnos.envel function. The object to be passed is an element of the resulting list. If the matrix has a column called "labels" then these are used as plot marks in the plot (useful for movement residuals).

ylim a numeric vector of length 2 giving the minimum and maximum values for the y-axis. This parameter is very similar to that described in par but has slightly different implementation (very slight).

... other graphical parameters to be passed to the plot function.

Details

This function is simple but care will need to be exercised in its use. See the example for best instruction and see Foster and Bravington (2009) for details about what the residuals are.

Author(s)

Scott D. Foster

References


See Also

RMC to estimate the Markov model, and diagnos and diagnos.envel to calculate residuals and simulation envelopes.

Examples

#produces Figure 1 of Foster and Bravington (2009)
#fit RMC model to dataEG1
fm.est <- RMC.mod( states=dataEG1[,2], chain.id=dataEG1[,1], X=dataEG1[,3])
#calculate the simulation envelope (many fewer samples than Foster and Bravington (2009)
B<50
temp.est <- diagnos.envel( obs.states=dataEG1[,2], chain.id=dataEG1[,1], X=dataEG1[,3,drop=FALSE], fit.fm.est, B
#plot the patch residuals for the second state and the movement residuals
par(mfrow=c(1,2))
hrplot( temp.est[["patch"]][[2]], ylab="Patch Residuals - Normal Quantiles", xlab="Normal Quantiles", main="Stationary data")
hrplot( temp.est$movement, ylab="Movement Residuals - Normal Quantiles", xlab="Normal Quantiles", main="Stationary data")
MVfill

Fill in missing values via a single imputation from the fitted model.

Description

_inserts the fitted probability of observing the chained data used in estimation for the model (fm). To be used when the outcome variable is used as a covariate at subsequent stages of an analysis. The fitted probability is conditional on the nearest observed value in the chain._

Usage

MVfill( fm, states=NULL, chain.id=NULL, X=NULL)

Arguments

- **fm**: a fitted model for the outcome that is required to be filled. Must be the result of a call to RMC.mod
- **states**: the outcome vector used to estimate the fitted model. Must be sequentially ordered within chains
- **chain.id**: a vector indicating which states belong to which chains
- **X**: the design matrix for the model used to create fm. Column ordering must match that provided to fm and row ordering must match that in states (and chain.id)

Value

_a matrix with number of columns equal to the number of categories of the outcome variable. Each column contains the fitted probability of observing the chained data used in estimation for the model (fm), conditional on the nearest observed value in the chain._

Author(s)

Scott D. Foster

References


Examples

```r
#form data with NAs in the outcomes
dataEG2.NA <- dataEG2
dataEG2.NA[sample(1:nrow(dataEG2.NA), 1000), "state"] <- NA #specify lots of NAs
#fit a model using dataEG2
fm.est2.NA <- RMC.mod( states=dataEG2.NA[,2], chain.id=dataEG2.NA[,1], X=dataEG2.NA[,-(1:2)])
#impute the missing values. The result will be a matrix
imputedData <- MVfill( fm.est2.NA, states=dataEG2.NA[,2], chain.id=dataEG2.NA[,1], X=dataEG2.NA[,3:4])
print( head( imputedData, 20))
```
Description

RMC is a package that fits and predicts reversible Markov models with a particular parameterisation described in Foster et al (2009). The core work-horse of the estimation routines is the function \texttt{rmcNmod}, but also see \texttt{MVfill} for (single) imputing of associated chained covariates, and \texttt{rmcNpred} for prediction of the local area stationary distribution.

Also contained in the RMC package is a bunch of methods that provide graphical diagnostics for this class of models (see Foster and Bravington 2010). This is performed through the functions \texttt{diagnos} for calculation of residuals, \texttt{diagnos.envel} for calculation of residuals and simulation envelopes, and \texttt{hrplot} for subsequent plotting.

Author(s)

Scott D. Foster

References


See Also

\texttt{diagnos, diagnos.envel, examplesForDiagnostics, hrplot, MVfill, RMC.mod, RMC.pred}.

\texttt{RMC.mod} \hspace{1cm} \textit{Estimation of categorical discrete-time non-stationary Markov chain models with simple parameterisation.}

Description

Estimation of categorical Markov chain models whose parameterisation is based on a simple reversible Markov model and that can be extended to non-stationary cases. The model is parameterised by two vectors of parameters: one describing the probability of moving from each state (phi) and the other describing the probability of moving into each state given that a movement will occur (pi). Non-stationary models are incorporated by letting each of these vectors depend on covariates.

Usage

\texttt{RMC.mod( states, chain.id=NULL, X=NULL, phi.id=NULL, pi.id=NULL, vcov=FALSE, inits=NULL, contr=list(}
Arguments

- **states**: observed ordered chained data. If there are multiple chains then chains are stacked on top of each other. Argument must be supplied.
- **chain.id**: vector (length matches states) of identifiers for the individual chains. If NULL then it is assumed that all observations form a single chain.
- **X**: design matrix (covariates) for the two vectors of probabilities. If NULL then X is assumed to contain an intercept term only. If not NULL then model will depend on phi.id and pi.id matrices (see below). X must be of dimensions nrow(X)=length(states) and ncol(X)=number of covariates. Typically will be created with a call to model.matrix
- **phi.id**: indicator matrix of zeros and ones showing which covariates to include in the model for which element of phi (zero means not included and one means included). Each element of phi corresponds to the probability of moving from an observed state. phi.id must be of dimensions nrow(phi.id)=ncol(X)=number of covariates and ncol(phi.id)=number of states. If NULL then all covariates are included. Covariates are included via a logistic model for each element of phi
- **pi.id**: indicator vector of zeros and ones showing which covariates to include in the model for all elements of pi (zero means not included and one means included). Each element of pi correspond to the probability of moving to that state given that a movement will occur. pi.id must have length equal to the number of covariates and indicates if that covariate is included in the model for pi. Covariates are included via the additive logistic transformation (Aitchison 1982)
- **vcov**: boolean indicating if the variance matrix of the parameter estimates should be calculated. TRUE indicates that it is calculated
- **inits**: initial values for the parameters. Must be of appropriate length and ordered as phi parameters and the pi parameters. If NULL then initial values are assumed to be zero. The ordering of this vector is: phi parameters for category 1, category 2, etc followed by pi parameters for transformed category 1, transformed category 2, etc.
- **contr**: list containing control values for the optimisation procedure. maxit specifies the maximum number of iterations before optimisation is stopped. epsg, epsf and epsx give the stopping tolerances for gradients, relative function and estimates respectively
- **quiet**: boolean indicating if any output is wanted. TRUE indicates that output is generated
- **penalty**: experimental argument for an optional quadratic penalty on the parameters. A non-zero value indicates that the sum of the squared parameters must be less than or equal to the value. A value of zero indicates no penalty and is the default.

Details

The observed chained categorical data (in argument states) is modelled according to that described in Foster et al (2009). The Markov process is assumed to be parameterised by two vectors, phi and pi. The phi parameters indicate the probability of moving from each state and the pi probabilities prescribe the probability of moving to each state given that a move will occur. This process is
reversible if the parameters do not change within a chain. The probabilities are allowed to vary within a chain by specifying these two vectors of probabilities as functions of covariates (possibly index number).

Since the model has simple form then the stationary distribution is known (up to normalisation constant) and hence, the (log-)likelihood is calculated exactly.

Optimisation is performed using a quasi-Newton method implemented in the LBFGS code from the ALGLIB website (see references). First derivatives for the optimisation are obtained using automatic differentiation (Griewank 2001) using the CppAD tool for C++ (Bell 2007). This saves an awful lot of mucking around with derivative free methods and increases speed. If you do not already use automatic differentiation then you may want to look into it.

Value

Upon successful completion the function returns

pars the parameter estimates ordered as phi parameters and then pi parameters. The ordering of this vector is: phi parameters for category 1, category 2, etc followed by pi parameters for transformed category 1, transformed category 2, etc.

like the maximised log-likelihood

scores the gradients calculated at the estimates. Ordered to match the pars vector

vcov the variance matrix of the estimates if vcov==TRUE and NULL if vcov==FALSE

conv the convergence code from the quasi-Newton optimiser

time the time taken to perform the fit

niter the number of iterations required by the optimiser

stuff quite literal: stuff used for model specification and optimisation. Generally not of use to the user

Author(s)

Scott D. Foster

References


ALGLIB http://www.alglib.net/ (accessed June 2008)


See Also

RMC.pred for predicting the stationary distribution at arbitrary combinations of covariates. diagnos and diagnos.envl for graphical diagnostic methods for models of class RMC.

Examples

# estimate a model for the stationary example data, dataEG1
fm.est1 <- RMC.mod( states=dataEG1[,2], chain.id=dataEG1[,1], X=dataEG1[,3])
# estimate a model for the non-stationary example data, dataEG2
fm.est2 <- RMC.mod( states=dataEG2[,2], chain.id=dataEG2[,1], X=dataEG2[,-(1:2)])

RMC.pred Prediction of local area stationary distribution for an arbitrary number of covariate combinations.

Description

Predicts the "local area stationary distribution" (with standard errors), and the average of all predictions (with standard errors). The latter prediction and the standard error can be taken to be an areal prediction if the number of points in the area is large and the covariance is not too high.

Usage

RMC.pred( fit, fit2=NULL, pts)

Arguments

fit an object resulting from an RMC fit. This model must relate to the set of observations whose marginal distribution is required.

fit2 an object resulting from an RMC fit whose outcomes are covariates for the outcomes in the object fit. See Foster et al. (2009) for details on the process.

pts a data matrix whose covariates must match those in the RMC objects fit and fit2. This matrix defines the points where the predictions are to occur.

Details

Predictions are made at each specified point by calculating the stationary distribution of the transition matrix for that combination of covariates. The variance, due to parameter uncertainty, of these prediction is obtained using the delta approximation. For predictions of a univariate model these derivatives are found using automatic differentiation (see Griewank 2001) implemented using the CppAD tool (Bell 2007). For predictions of a bivariate model the derivatives are found using finite differences and hence this procedure can run a little slow (well, a lot slow). If this causes problems then a more sophisticated implementation can be considered.

Global (or areal if the points come from a contiguous region of space) predictions are also calculated as the simple average of the predicted points. The variance of this average can be used as a prediction variance if the number of prediction points is large and the covariance between them is not too severe.
Predictions for univariate chained data are obtained by specifying a single model to the "fit" argument (keeping the "fit2" argument NULL). Predictions for the second variable of bivariate chained data are obtained by specifying the target univariate model in the "fit" argument, and the non-target univariate model in "fit2". This methodology was developed to predict fauna types where geomorphology (also chained data) was used as a covariate in the fauna model. Details are given in Foster et al. 2009.

Value

area the average of the predictions. Can be used as an areal prediction
pts point predictions for each of the rows in the prediction matrix
vcov variance and covariance matrix of the areal predictions. This is only calculated if the parameter estimate’s variance for fit (and fit2 if not null) are present

Author(s)

Scott D. Foster

References


See Also

`RMC.mod` to estimate the Markov model, `mvfill` to impute any missing values in a particular sub-set of covariates, and `sim.chain` to simulate chained data.

Examples

```r
#fit model to non-stationary data including all covariates
fm.est1 <- RMC.mod( states=dataEG2[,2], chain.id=dataEG2[,1], X=dataEG2[,3:4], vcov=TRUE)
#estimate the model
#perform predictions
pred1 <- RMC.pred( fit=fm.est1, fit2=NULL, pts=dataEG2[,3:4])
tmp <- cbind( table( dataEG2,"state") / nrow( dataEG2), pred1$area, sqrt( diag( pred1$vcov)))
colnames( tmp) <- c("observed", "predicted", "se")
rownames( tmp) <- paste( "cat",1:fm.est1$stuff$n.cats, sep="_")
print( tmp)
###simulate and predict from bivariate non-stationary chained data
n.cats1 <- fm.est1$stuff$n.cats; n.cats2 <- 5; n.covars <- 2; n.covars2 <- n.covars*n.cats1 + n.covars
#fill in missing values (if any) for the previous outcomes that are now covariates
level1Probs <- mvfill( fm.est1, states=dataEG2[,2], chain.id=dataEG2[,1], X=dataEG2[,3:4])
#setting up design matrix -- note the order, it is important for RMC.pred
```
X2 <- matrix( rep( dataEG2[,3:4], n.cats1+1), nrow=nrow( dataEG2))
for( ii in 1:n.cats1)
  X2[ii*n.covars+1:n.covars] <- X2[ii*n.covars+1:n.covars] * rep( level1Probs[ii], n.covars)
colnames(X2) <- paste( c( "const", "rand"), rep("",1:n.cats1), each=2, sep="")
#specify parameter values, note that parameterisation gives state 1 as a reference state for all categories. This
gamma <- matrix( rnorm( n.covars*n.cats2, sd=0.5), nrow=n.covars2, ncol=n.cats2) #initial design matrix
for( ii in 1:n.cats2)
  gamma[sample( 2:n.covars2, sample(n.covars2-1:4, 1)), ii] <- 0
beta <- matrix( rnorm( n.covars*n.cats2, sd=0.5), nrow=n.covars2, ncol=n.cats2)
beta[n.covars+1:n.covars,] <- 0
beta[,n.cats2] <- 0
#simulate chains
chains2 <- sim.chain( n.chains=5, n.obs=rep( 1000, 5), n.cats=n.cats2, n.covars=n.covars2, beta=beta, gamma=gamma)
#simulate the chained categorical data
#specify RMC model to be estimated
my.phi.id <- ifelse( gamma!=0, 1, 0) #model controlling matrix
my.pi.id <- apply( beta, FUN=function(x){if( any( x!=0) ) 1 else 0}, MARG=1) #model controlling matrix
#fit model
fm.est2 <- RMC.mod( states=chains2[,2], chain.id=chains2[,1], X=X2, phi.id=my.phi.id, pi.id=my.pi.id, vcov=TRUE)
#estimate the model
#perform predictions
pred2cond <- RMC.pred( fit=fc.est2, fit2=FALSE, pts=X2)
pred2marg <- RMC.pred( fit=fc.est2, fit2=fc.est1, pts=dataEG2[,3:4])
#check against empirical value
tmp <- cbind( table( chains2[,"state"])/ nrow( chains2), pred2cond$area, sqrt( diag( pred2cond$vcov)), pred2marg$area)
colnames( tmp) <- c("observed","conditional prediction", "conditional se", "marginal prediction", "marginal se")
rownames( tmp) <- c("category", 1:n.cats2, sep="_")
print( tmp)

---

**Simulate Markov chain data from a Markov model.**

**Description**

Simulate chained data from a simple reversible Markov model (see Foster et al 2009 for details). Simulates stationary and non-stationary data, the later is formed by defining the transition matrix as a function of covariates.

**Usage**

```r
sim.chain( n.chains = 5, n.obs = rep( 100, n.chains), n.cats = 3, n.covars = 1, beta = NULL, gamma = NULL)
```

**Arguments**

- `n.chains` scalar specifying the number of chains to simulate. The default value is 5.
- `n.obs` vector of length `n.chains`. Indicates the number of observations per chain. Default is `rep( 100, n.chains)` for 5 chains of 100 observations.
- `n.cats` scalar specifying the number of categories in the chain. Default is 3 states.
The number of covariates that affect the transition matrix. The constant must be considered to be one of these covariates. Default is n.covars=1 for constant term only (stationary chain).

Elements of the matrix of coefficients for the probability of moving into each state, used to partially define the transition matrices (also need gamma). These values are transformed via the additive logistic transform. The dimension of the beta matrix must be nrow=n.covars and ncol=n.cats with rows indexing the covariates and columns indexing the transformed probabilities (transformed from a simplex with the additive logistic transform). The first column of this matrix must be zero, reflecting that the last category’s transition probability is constrained.

Elements of the matrix of coefficients for the probability of moving from any particular state, used to partially define the transition matrices (also need beta). These parameters are transformed using the logistic transform. The dimension of the gamma matrix must be nrow=n.covars and ncol=n.cats with rows indexing covariates and columns indexing states.

The design matrix for the covariates. If NULL (the default) then the design matrix is filled with a constant and random (normal) variables. The design matrix must be numeric with all factors suitably converted into dummy variables and so on.

The observed chained categorical model is defined according to Foster et al (2009). The Markov process is assumed to be parameterised by two vectors, phi and pi. The phi parameters indicate the probability of moving from each state and the pi probabilities prescribe the probability of moving to each state given that a move will occur. This process is reversible if the parameters do not change within a chain. The probabilities are allowed to vary within a chain by specifying these two vectors of probabilities as functions of covariates (possibly index number).

The function returns a matrix with columns for the chain identification (values from 1 to n.chains), the simulated chained data, and a column for each of the scaled X matrix.

Scott D. Foster


# simulate 3 chains with 4 states of length 1500. There is one (constant) covariate. The beta parameters are for the pi probabilities and the gamma parameters are for the phi probabilities (note this is the same parameters used to generate data).

```r
chain <- sim.chain( n.chains=3, n.obs=rep( 1500, 3), n.cats=4, n.covars=1, beta=c(0,0.3,-0.3,0), gamma=c(0.5,0.2))
```
# Plot the first 100 observations of each chain
par(mfrow=c(3,1))
for(ii in 1:3){
tmpDat <- chain[,"chain"]==ii,
plot(1:100, tmpDat[1:100,"state"], main=paste("Chain", ii), ylab="State", xlab="Index", type='b', pch=20)
abline(h=1:4, lty=2)
}

simRandWalk

Simulate a continuous auto-regressive process.

Description

Simulates an autoregressive process for a specified number of sets of observations. The first observation in each chain is drawn from an independent normal and subsequent observations are drawn from another normal with mean equal to the last observation.

Usage

simRandWalk( nc=5, ni=rep(1000, nc), init.var=1, seq.var=0.1)

Arguments

nc the number of chains to simulate
ni the length of each chain (must be a vector of length nc)
init.var the variance to use for the initial random number generation (distribution will have mean zero)
seq.var the variance to use in the sequential simulation

Details

The autoregressive process is simulated for each of nc chains. The first element of each chain is simulated from a normal with zero mean and variance init.var. The subsequent values are simulated from a random draw from a normal with mean equal to the previous observation and variance seq.var.

Value

A matrix with sum(ni) rows and 2 columns. The first column has elements 1:nc and indicates the chain the observation belongs to. The second column contains the random values.

Author(s)

Scott D. Foster
Examples

```r
ni <- c(30, 300, 3000)
simDat <- simRandWalk(nc=3, ni=ni, init.var=1, seq.var=0.1)
par(mfrow=c(1,3))
plot(1:ni[1], simDat[1:ni[1],2], type='b', pch=20, ylab="Random Variable", xlab="Index", main="Chain 1")
plot(1:ni[2], simDat[ni[1]+1:ni[2],2], type='b', pch=20, ylab="Random Variable", xlab="Index", main="Chain 2")
plot(1:ni[3], simDat[sum(ni[1:2])+1:ni[3],2], type='b', pch=20, ylab="Random Variable", xlab="Index", main="Chain 3")
```
Index

*Topic **misc**
  dataEG1, 2
  dataEG2, 2
  dataEG3movement, 2
  dataEG3patch, 3
  dataEG4, 3
  diagnos, 4
  examplesForDiagnostics, 6
  hrplot, 8
  MVfill, 10
  RMC, 11
  RMC.mod, 11
  RMC.pred, 14
  sim.chain, 16
  simRandWalk, 18

  dataEG1, 2
  dataEG2, 2
  dataEG3movement, 2
  dataEG3patch, 3
  dataEG4, 3
  diagnos, 2, 3, 4, 6, 9, 11, 14
  diagnos.envel, 2, 3, 6, 9, 11, 14

  examplesForDiagnostics, 6, 11
  hrplot, 2, 3, 5, 6, 8, 11
  MVfill, 10, 11, 15

  RMC, 9, 11
  RMC.mod, 5, 6, 10, 11, 11, 15
  RMC.pred, 11, 14, 14

  sim.chain, 6, 15, 16
  simRandWalk, 18