Package ‘march’

August 18, 2019

Title Markov Chains

Description Computation of various Markovian models for categorical data including homogeneous Markov chains of any order, MTD models, Hidden Markov models, and Double Chain Markov Models.

Version 3.1

Date 2019-07-29

Author Ogier Maitre, with contributions from Andre Berchtold, Kevin Emery and Oliver Buschor

Maintainer Andre Berchtold <andre.berchtold@unil.ch>

Depends R (>= 3.0.2), methods

License GPL-2

LazyData true

RoxygenNote 6.1.1

Encoding UTF-8

Repository CRAN

Repository/R-Forge/Project march

Repository/R-Forge/Revision 112

Repository/R-Forge/DateTimeStamp 2019-07-29 14:41:18

Date/Publication 2019-08-18 10:30:02 UTC

NeedsCompilation no

R topics documented:

`march-package` *Computation of Markovian models for categorical data*

Description

This package is dedicated to the computation of various Markovian models for categorical data including the independence model, homogeneous Markov chains of any order, the Mixture Transition Distribution (MTD) model for the approximation of high-order homogeneous Markov chains, Hidden Markov Models (HMMs) and Double Chain Markov Models (DCMMs).

Author(s)

Ogier Maitre, Andre Berchtold

References

Berchtold A, Raftery AE (2002) The Mixture Transition Distribution Model for High-Order Markov Chains and Non-Gaussian Time-Series. *Statistical Science* 17(3), 328-356.

Berchtold A (2002) High-order extensions of the Double Chain Markov Model. *Stochastic Models* 18, 193-227.

See Also

`march.Model-class`, `march.Dataset-class`.

---

march.AIC — *Compute Akaike Information Criterion (AIC). The AIC (Akaike Information Criterion) is computed for a given `march.Model-class` according to the data used during construction.*

Description

Compute Akaike Information Criterion (AIC).

The AIC (Akaike Information Criterion) is computed for a given `march.Model-class` according to the data used during construction.

Usage

```
march.AIC(model)
```

Arguments

`model`	The model for which the AIC has to be computed.

Value

The number of parameters of the given model and its AIC.

Author(s)

Ogier Maitre

Examples

```
indepModel <- march.indep.construct(pewee)
march.AIC(indepModel)
```

march.BIC — *Compute Bayesian Information Criterion (BIC).*

Description

The BIC (Bayesian Information Criterion) is computed for a given `march.Model-class` according to the data used during construction.

Usage

```
march.BIC(model)
```

Arguments

`model` The model for which the BIC has to be computed.

Value

The number of parameters of the given model and its BIC.

Author(s)

Ogier Maitre

Examples

```
indepModel <- march.indep.construct(pewee)
march.BIC(indepModel)
```

march.Dataset-class — *Dataset for march package.*

Description

This class contains several discrete-valued time series, in a dataset. It contains for each sequence, its length and weights.

Details

The internal representation uses factor-like representation. The integer values correspond to the words stored into the dictionary vector. Therefor, they are in the interval [1,K].

@section Slots:

`yRaw`: A matrix of `character` string, describing the content of the original dataset or file, if any.

`y`: A list of vector of `integer` representing the each discrete-valued time series of the dataset, as can be used by the models.

`T`: A vector of `integer` values representing the length of each sequence.

`weights`: A vector of `numeric` values representing the weight of each sequence.

`K`: A `integer` value representing the number of possible ouput and the number of words stored into the dictionary.

`N`: A `integer` value representing the number of sequence.

`Dictionary`: A vector of `character` string representing the translation between the yRaw and y data. Each character string is stored according to the integer which represents it into y.

`cov`: A matrix of `integer` representing the covariates.

`Kcov`: A vector of `integer` representing the number of possible output for each covariate.

`Ncov`: A `integer` value representing the number of covariates.

@seealso `march.dataset.loadFromFile`, `march.dataset.loadFromDataFrame` @author Ogier Maitre

`march.dataset.h.extractSequence` *Extract a sequence from a dataset.*

Description

Extract a sequence from a dataset.

Usage

```
march.dataset.h.extractSequence(y, i)
```

Arguments

`y`	A sequence of integers.
`i`	The number of observations to keep.

Author(s)

Ogier Maitre

march.dataset.loadFromDataFrame

Construct a dataset from a data.frame or a matrix.

Description

The function creates a `march.Dataset-class` from a *dataframe* or a *matrix*, where each row (resp. column) represents an independent data series when *MARGIN* is 2 (resp. 1).

Usage

```
march.dataset.loadFromDataFrame(dataframe, MARGIN = 2, weights = NA,
  missingDataRep = NA, covariates = NULL)
```

Arguments

`dataframe`	A `data.frame` containing the dataset.
`MARGIN`	The dimension of the matrix/data.frame that contains the sequences and of the covariates (resp 1 for the column, 2 for the rows).
`weights`	If specified, contains the weight of each sequence.
`missingDataRep`	If specified, the symbol representing a missing data.
`covariates`	If specified, a three dimensional array of integers, representing the covariates. The data for the i-th covariates should be in [, , i]. If the data are column-wise (respectively row-wise), each table of covariates should be column-wise (respectively row-wise). If we only have one covariate, we can simply pass a two-dimensional array. The covariates should be coded as integers from 1 to the number of possible outputs.

Value

A `march.Dataset-class` object containing the data contructed from the matrix or data.frame.

Author(s)

Ogier Maitre

Examples

```
# Create a march dataset from the sleep_df dataframe included in the march package.
sleep <- march.dataset.loadFromDataFrame(sleep_df, MARGIN = 2,
                            weights = NA, missingDataRep = NA)

# Each row of sleep_df contains the data for one subject, so MARGIN was set to 2.

# Most of the subjects have been observed during 7 consecutive years,
# but some subjects have been observed for only 5 or 6 years.
# To load only the first 5 observations of each subject:
```

```
sleep.5 <- march.dataset.loadFromDataFrame(sleep_df[,1:5], MARGIN = 2 ,
                            weights = NA, missingDataRep = NA)

# The sleep data are not weighted.
# To add a weighting variable taking value 1.5 for the 500 first subjects
# and value 0.5 for the 500 next:
weighting <- rep(1.5,1000)
weighting[501:1000] <- rep(0.5,500)
sleep.w <- march.dataset.loadFromDataFrame(sleep_df, MARGIN = 2,
                           weights = weighting, missingDataRep = NA)

# We add two covariates to the sleep data. The first is the sex of the subject
# (1 for male, 2 for female), and the second is the age range (1 for younger
# than 40, 2 for older than 40). We suppose that the first 250 sequences
# represent men older than 40, the next 250 sequences men younger than 40,
# the next 250 women younger than 40 and the last 250 women older than 40.
# We build the two tables of covariates and bind them together to obtain a
# three dimensional array that can be handled by MARCH.
covariates.sex<-rbind(matrix(1,500,7),matrix(1,500,7))
covariates.age<-rbind(matrix(1,250,7), matrix(2,250,7), matrix(1,250,7),
                    matrix(2,250,7))
covariates<-array(0,c(1000,7,2))
covariates[ , ,1]<-covariates.sex
covariates[ , ,2]<-covariates.age
# We build a MARCH dataset object containing these covariates.
sleep.covariates<-march.dataset.loadFromDataFrame(sleep_df,covariates=covariates)
```

`march.dataset.loadFromFile` *Load a dataset from a file.*

Description

The function loads a dataset from a text file, where each row (resp. column) represents a data series when *MARGIN* is 2 (resp. 1), using the character *sep* as attribute separator. Each data sequence should be stored in a given column, (resp. row).

Usage

```
march.dataset.loadFromFile(filename, MARGIN = 2, sep = ",",
  weights = NA)
```

Arguments

`filename`	The complete path to the text file containing the dataset.
`MARGIN`	The dimension of the extracted data.frame that contains the sequences (resp 1 for the column, 2 for the rows).
`sep`	A caracter used as element separator on a line.
`weights`	If specified, contains the weight of each sequence.

Value

a `march.Dataset-class` object containing the data from the file found at *filename*, using separator *sep*.

Author(s)

Ogier Maitre #'

`march.Dcmm-class` *A Double Chain Markov Model (DCMM).*

Description

This class describes a Double Chain Markov Model (DCMM) represented by Pi, the probability distributions of the first hidden states; by A, the transition matrix between hidden states; by RB, the transition matrix between sucessive output. march. Dcmm extends `march.Model-class` class and therefore inherits its slots.

Details

The model used here is described in :

- Berchtold, A.: The Double Chain Markov Model. Commun. Stat., Theory Methods 28 (1999), pp. 2569-2589
- Berchtold, A.: High-order extensions of the Double Chain Markov Model. Stochastic Models 18 (2002), pp. 193-227.

Slots

`Pi`**:** A 3D matrix of `numeric` representing the probability distribution of the first hidden state.

`A`**:** A matrix of `numeric` representing the transition matrix between hidden states.

`RB`**:** A 3D matrix of `numeric` representing the transition matrix between successive output, in a reduced form.

`M`**:** An `integer` value representing the number of hidden state.

`orderVC`**:** An `integer` value representing the order of the visible Markov chain.

`orderHC`**:** An `integer` value representing the order of the hidden Markov chain.

`Amodel`**:** A vector of `character` string representing the modeling of the hidden transition matrix (complete, mtd or mtdg)

`Cmodel`**:** A vector of `character` string representing the modeling of the visible transition matrix (complete, mtd or mtdg)

See Also

`march.dcmm.construct`, `march.Model-class`.

march.dcmm.construct *Construct a double chain Markov model (DCMM).*

Description

Construct a march.Dcmm-class object, with visible order *orderVC*, hidden order *orderHC* and *M* hidden states, according to a march.Dataset-class. The first *maxOrder-orderVC* elements of each sequence are truncated in order to return a model which can be compared with other Markovian model of visible order maxOrder. The construction is performed either by an evolutionary algorithm (EA) or by improving an existing DCMM. The EA performs *gen* generations on a population of *popSize* individuals. The EA behaves as a Lamarckian evolutionary algorithm, using a Baum-Welch algorithm as optimization step, running until log-likelihood improvement is less than *stopBw* or for *iterBw* iterations. Finally only the best individual from the population is returned as solution. If a seedModel is provided, the only step executed is the optimization step, parameters related to the EA does not apply in this case.

Usage

```
march.dcmm.construct(y, orderHC, orderVC, M = 2, gen = 5,
  popSize = 4, maxOrder = orderVC, seedModel = NULL, iterBw = 2,
  stopBw = 0.1, Amodel = "mtd", Cmodel = "mtd", AMCovar = 0,
  CMCovar = 0, AConst = FALSE, pMut = 0.05, pCross = 0.5)
```

Arguments

`y`	the dataset from which the Dcmm will be constructed march.Dataset-class.
`orderHC`	the order of the hidden chain of the constructed Dcmm.
`orderVC`	the order of the visible chain of the constructed Dcmm (0 for a HMM).
`M`	the number of hidden state of the Dcmm.
`gen`	the number of generation performed by the EA.
`popSize`	the number of individual stored into the population.
`maxOrder`	the maximum visible order among the set of Markovian models to compare.
`seedModel`	a model to optimize using Baum-Welch algorithm.
`iterBw`	the number of iteration performed by the Baum-Welch algorithm.
`stopBw`	the minimum increase in quality (log-likelihood) authorized in the Baum-Welch algorithm.
`Amodel`	the modeling of the hidden transition matrix (mtd, mtdg or complete)
`Cmodel`	the modeling of the visible transition matrix (mtd, mtdg or complete)
`AMCovar`	vector of the size Ncov indicating which covariables are used into the hidden process (0: no, 1:yes)
`CMCovar`	vector of the size Ncov indicating which covariables are used into the visible process (0: no, 1:yes)

AConst	logical, indicating whether or not the hidden transition matrix is diagonal constraint
pMut	mutation probability for the evolutionary algorithm
pCross	crossover probability for the evolutionary algorithm

Value

the best march.Dcmm-class constructed by the EA or the result of the Baum-Welch algorithm on *seedModel*.

Author(s)

Emery Kevin

See Also

march.Dcmm-class, march.Model-class, march.Dataset-class.

Examples

```
# Construct a 2 hidden states DCMM for the pewee data
# with hidden order set to 2 and visible order set to 1.
# The estimation procedure uses both the evolutionary algorithm (population size 2,
# one generation) and the Bauw-Welch algorithm (one iteration).
## Not run: march.dcmm.construct(y=pewee,orderHC=2,
                              orderVC=1,M=2,popSize=2,gen=1,iterBw=1,stopBw=0.0001)

# Same as above, but the DCMM is replaced by a HMM (the visible order OrderVC is set to zero).
HMM<-march.dcmm.construct(y=pewee,orderHC=2,orderVC=0,M=2,popSize=2,gen=1,iterBw=1,stopBw=0.0001)

# A first model is computed using both EA and Baum-Welch algorithms.
# The previous model is improved through two rounds of Baum-Welch optimization.
models <- list()
models[[length(models)+1]] <- HMM
models[[length(models)+1]] <- march.dcmm.construct(y=pewee,seedModel=models[[1]],
                                                   orderVC=0,iterBw=10,stopBw=0.001)
models[[length(models)+1]] <- march.dcmm.construct(y=pewee,seedModel=models[[2]],
                                                   orderVC=0,iterBw=10,stopBw=0.0001)
# Show performance indicators (ll, number of independent parameters,
# BIC and AIC) for all computed models.
#r <- do.call(rbind,lapply(models,march.summary))
#print(r)

# Construct a three hidden states, first-order HMM (hence OrderVC=0) for the sleep data.
# By setting gen=1 and popSize=1, the estimation procedure uses only the Baum-Welch algorithm.
HMM <- march.dcmm.construct(pewee,orderHC=1,orderVC=0,M=2,gen=1,popSize=1,iterBw=10,stopBw=0.0001)
## End(Not run)
```

march.dcmm.viterbi — *Viterbi algorithm for a DCMM model.*

Description

Viterbi algorithm for a DCMM model.

Usage

```
march.dcmm.viterbi(d)
```

Arguments

d	The march.Dcmm-class on which to compute the most likely sequences of hidden states.

Value

A list of vectors containing the most likely sequences of hidden states, considering the given model for each sequence of the given dataset.

Author(s)

Kevin Emery

Examples

```
set.seed(327)
# Computation of a DCMM model
## Not run: model <- march.dcmm.construct(y=pewee,orderHC=1,orderVC=1,M=2,popSize=1,gen=1)
# Extraction of the best sequence of hidden states using the Viterbi algorithm.
bestSeq <- march.dcmm.viterbi(model)
print(bestSeq)
## End(Not run)
```

march.Indep-class — *An independence model.*

Description

This class describes an independence model, represented by the probability distribution *indP* of each event and the number of data used to compute each member of the probability distribution. march.Indep inherits from march.Model-class and therefore inherits its slots.

Slots

`indP`: A vector of `numeric` representing the model probability distribution.

`indC`: A vector of `integer` representing the number of data used to compute each member of the probability distribution.

See Also

`march.indep.construct`, `march.Model-class`.

---

march.indep.construct *Construct an independence model (zero-order Markov chain).*

Description

Construct a `march.Indep-class` model from a given `march.Dataset-class`, the first *maxOrder* elements of each sequence being truncated in order to return a model which can be compared with other Markovian models of visible order maxOrder.

Usage

```
march.indep.construct(y, maxOrder = 0)
```

Arguments

`y`	the `march.Dataset-class` from which construct the model.
`maxOrder`	the maximum visible order among the set of Markovian models to compare.

Value

The `march.Indep-class` constructed using dataset y and maxOrder.

Author(s)

Ogier Maitre

See Also

`march.Indep-class`, `march.Model-class`, `march.Dataset-class`.

Examples

```
# Build an independance model from the pewee data set.
model <- march.indep.construct(pewee)
print(model)
```

march.Mc-class *A Markov chain of order >= 1.*

Description

This class describes a Markov chain of order *order*, represented by matricess RC (transition matrix in reduced form) and RT (number of data points used to compute each transition). march.Mc extends `march.Model-class` class and therefore inherits its slots.

Slots

- `RC`: A matrix of `numeric` representing the reduced form of the transition matrix of the current Markov Chain.
- `order`: An `integer` representing the order of the current Markov Chain.
- `RT`: A matrix of `integer` representing the number of sample used to compute each transition row of the current RC matrix.

See Also

`march.mc.construct`, `march.Model-class`.

march.mc.construct *Construct an homogeneous Markov Chain.*

Description

A `march.Mc-class` object of order *order* is constructed from the dataset *y*. The first maxOrder-order elements of each sequence of the dataset are truncated in order to return a model which can be compared with other Markovian models of visible order maxOrder.

Usage

```
march.mc.construct(y, order, maxOrder = order)
```

Arguments

`y`	the `march.Dataset-class` from which the homogeneous Markov chain will be constructed.
`order`	the order of the constructed Markov Chain.
`maxOrder`	the maximum visible order among the set of Markovian models to compare.

Value

the `march.Mc-class` of order *order* constructed *w.r.t* the dataset *y* and maxOrder.

Author(s)

Ogier Maitre

See Also

march.Mc-class, march.Model-class, march.Dataset-class.

Examples

```
# pewee dataset is a data object of the march package in march.Dataset class format.
model <- march.mc.construct(pewee,2)

# print the reduced form of the transition matrix of the Markovian Model.
print(model@RC)
```

---

march.Model-class *A basic and virtual march model.*

Description

This class describe the basic and virtual model, that every model of the package will extend. This is a virtual class, which is not meant to be handled by user directly.

See Also

The classes that inherit from march.Model are : march.Indep-class, march.Mc-class, march.Mtd-class, march.Dcmm-class.

@section Slots:

ll: A numeric representing the log-likelihood for this model *w.r.t* its construction dataset.

y: The march.Dataset-class used to construct the model.

dsL: A numeric representing the number of sample used to construct the model.

nbZeros: A numeric representing the number of zeros created during model construction.

march.Mtd-class — *A Mixture Transition Distribution (MTD) model.*

Description

This class describes a Mixture Transition Distribution (MTD) model, represented by its transition matrix Q, its vector phi of lag parameters and its order. march.Mtd extends `march.Model-class` class and therefore inherits its slots. march.Mtd extends `march.Model-class` class and therefore inherits its slots.

Details

The model used here is described into :

- Raftery, A. E. A Model for High-Order Markov Chains. JRSS B 47(1985), pp. 528-539.
- Berchtold, A. Estimation in the mixture transition distribution model. Journal of Time Series Analysis, 22 (4) (2001), pp. 379-397

@section Slots:

`Q`: A matrix of `numeric` representing the transition matrix associated with the current MTD model.

`S`: A list of matrices of `numeric` representing the transition matrices between the covariates and the dependent variable

`phi`: A vector of `numeric` representing the vector of lag parameters.

`order`: An `integer` representing the order of the model.

See Also

`march.mtd.construct`, `march.Model-class`.

march.mtd.construct — *Construct a Mixture Transition Distribution (MTD) model.*

Description

A Mixture Transition Distribution model (`march.Mtd-class`) object of order *order* is constructed according to a given `march.Dataset-class` *y*. The first *maxOrder-order* elements of each sequence are truncated in order to return a model which can be compared with other Markovian models of visible order maxOrder.

Usage

```
march.mtd.construct(y, order, maxOrder = order, mtdg = FALSE,
  MCovar = 0, init = "best", deltaStop = 1e-04, llStop = 0.01,
  maxIter = 0)
```

Arguments

`y`	the dataset (`march.Dataset-class`) from which to construct the model.
`order`	the order of the constructed model.
`maxOrder`	the maximum visible order among the set of Markovian models to compare.
`mtdg`	flag indicating whether the constructed model should be a MTDg using a different transition matrix for each lag (value: *TRUE* or *FALSE*).
`MCovar`	vector of the size Ncov indicating which covariables are used (0: no, 1:yes)
`init`	the init method, to choose among *best*, *random* and *weighted*.
`deltaStop`	the delta below which the optimization phases of phi and Q stop.
`llStop`	the ll increase below which the EM algorithm stop.
`maxIter`	the maximal number of iterations of the optimisation algorithm (zero for no maximal number).

Author(s)

Ogier Maitre, Kevin Emery

See Also

`march.Mtd-class`, `march.Model-class`, `march.Dataset-class`.

Examples

```
# Build a 4th order MTD model from the pewee data set.
model <- march.mtd.construct(pewee,4)
print(model)

# Build a 3th order MTDg model from the pewee data set.
model <- march.mtd.construct(pewee,3,mtdg=TRUE)
print(model)
```

---

march.name — *march.Model name.*

Description

Generate a name for the march model contained in the given *object*.

Usage

```
march.name(object)
```

Arguments

`object`	contains the name of the model(Independence model, MTD,...).

Author(s)

Ogier Maitre & Andre Berchtold

Examples

```
# Compute a third-order homogeneous Markov Chain for the pewee data.
model <- march.mc.construct(pewee,3)

# Assign its name to the model.
march.name(model)
```

`march.name,march.Dcmm-method`
This method is called with the object "march.Dcmm" and provides it to the march.name function.

Description

This method is called with the object "march.Dcmm" and provides it to the march.name function.

Usage

```
## S4 method for signature 'march.Dcmm'
march.name(object)
```

Arguments

`object`	contains the name of the model.

`march.name,march.Indep-method`
This method is called with the object "march.Indep" and provides it to the march.name function.

Description

This method is called with the object "march.Indep" and provides it to the march.name function.

Usage

```
## S4 method for signature 'march.Indep'
march.name(object)
```

Arguments

`object`	contains the name of the model.

`march.name,march.Mc-method`
This method is called with the object "march.MC" and provides it to the march.name function.

Description

This method is called with the object "march.MC" and provides it to the march.name function.

Usage

```
## S4 method for signature 'march.Mc'
march.name(object)
```

Arguments

`object`	contains the name of the model.

`march.name,march.Mtd-method`
This method is called with the object "march.Mtd" and provides it to the march.name function.

Description

This method is called with the object "march.Mtd" and provides it to the march.name function.

Usage

```
## S4 method for signature 'march.Mtd'
march.name(object)
```

Arguments

`object`	contains the name of the model.

march.read — *Load a march.Model.*

Description

Load a march.Model from a file pointed by *filename* and check that the model is valid.

Usage

```
march.read(filename)
```

Arguments

`filename`	the path where load the mode

Value

the march.Model contained into the file pointed by filename if it exists and contains a valid model.

march.summary — *march.Model Summary.*

Description

Print key values for the current model.

Usage

```
march.summary(object, ...)
```

Arguments

`object`	can contain the results of any model computed using march
`...`	should indicate any additional parameter passed to the function

Author(s)

Ogier Maitre & Andre Berchtold

march.thompson *Thompson Confidence Intervals for a march.Model.*

Description

Compute the confidence intervals using Thompson's formula on a march.Model object. See Thompson SK (1987) Sample size for estimating multinomial proportions, American Statistician 41:42-46, for details.

Usage

```
march.thompson(object, alpha)
```

Arguments

`object`	the march.Model object on which compute the confidence intervals.
`alpha`	the significance level among : 0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.025, 0.02, 0.01, 0.005, 0.001, 0.0005, 0.0001.

Value

A list of half-length confidence intervals for each probability distribution of the considered model.

Author(s)

Ogier Maitre, Kevin Emery

Examples

```
# Compute a first-order homogeneous Markov Chain for the pewee data.
MC1 <- march.mc.construct(pewee,1)
# Display the transition matrix
print(MC1@RC)
# Compute the half-length 95% confidence interval for each row of the transition matrix.
march.thompson(MC1,alpha=0.05)

# Compute a third-order MTD model for the pewee data.
MTD3 <- march.mtd.construct(pewee,3)
# Display the model
print(MTD3)
# Compute the half-length 95% confidence interval for the vector of lags
# and for each row of the transition matrix.
march.thompson(MTD3,alpha=0.05)
```

march.thompson,march.Dcmm,numeric-method

This method is called with the object "march.Dcmm" and the aplha "numeric" and provides it to the march.thompson function.

Description

This method is called with the object "march.Dcmm" and the aplha "numeric" and provides it to the march.thompson function.

Usage

```
## S4 method for signature 'march.Dcmm,numeric'
march.thompson(object, alpha)
```

Arguments

object	contains the name of the model.
alpha	contains the Type I error

march.thompson,march.Indep,numeric-method

This method is called with the object "march.Indep" and the aplha "numeric" and provides it to the march.thompson function.

Description

This method is called with the object "march.Indep" and the aplha "numeric" and provides it to the march.thompson function.

Usage

```
## S4 method for signature 'march.Indep,numeric'
march.thompson(object, alpha)
```

Arguments

object	contains the name of the model.
alpha	contains the Type I error

march.thompson,march.Mc,numeric-method

This method is called with the object "march.Mc" and the aplha "numeric" and provides it to the march.thompson function.

Description

This method is called with the object "march.Mc" and the aplha "numeric" and provides it to the march.thompson function.

Usage

```
## S4 method for signature 'march.Mc,numeric'
march.thompson(object, alpha)
```

Arguments

`object`	contains the name of the model.
`alpha`	contains the Type I error

march.thompson,march.Mtd,numeric-method

This method is called with the object "march.Mtd" and the aplha "numeric" and provides it to the march.thompson function.

Description

This method is called with the object "march.Mtd" and the aplha "numeric" and provides it to the march.thompson function.

Usage

```
## S4 method for signature 'march.Mtd,numeric'
march.thompson(object, alpha)
```

Arguments

`object`	contains the name of the model.
`alpha`	contains the Type I error

`march.write` *Save a march.Model*

Description

Save a march.Model into a file pointed by *filename*. The save will fails if the file already exists unless force has been set to TRUE.

Usage

```
march.write(filename, object, force = FALSE)
```

Arguments

`filename`	a path to the file where to write the model (absolute or relative to the current directory).
`object`	the model to write.
`force`	if TRUE and if the file pointed by the filename path already exists, overwrite it. @return invisible TRUE if the model has been written into the file pointed by filename, invisible FALSE otherwise.

`pewee` *Song of the Wood Pewee (march dataset format)*

Description

This dataset contains a sequence of 1327 successive observations of the wood pewee song. This song consists in three different phrases numbered from 1 to 3.

Usage

```
pewee
```

Format

A march dataset.

Source

Craig (1943)

References

Craig, W. (1943) *The Song of the Wood Peewee*; University of the State of New York: Albany.

pewee_df — *Song of the Wood Pewee (data frame format)*

Description

This dataset contains a sequence of 1327 successive observations of the wood pewee song. This song consists in three different phrases numbered from 1 to 3.

Usage

```
pewee_df
```

Format

A data frame.

Source

Craig (1943)

References

Craig, W. (1943) *The Song of the Wood Peewee*; University of the State of New York: Albany.

pewee_t — *Song of the Wood Pewee (text format)*

Description

This dataset contains a sequence of 1327 successive observations of the wood pewee song. This song consists in three different phrases numbered from 1 to 3.

Usage

```
pewee_t
```

Format

A text file.

Source

Craig (1943)

References

Craig, W. (1943) *The Song of the Wood Peewee*; University of the State of New York: Albany.

`sleep` *Sleep disorders (march dataset format)*

Description

This dataset contains 1000 sequences of 5 to 7 successive observation of the level of sleep disorders. Each sequence corresponds to a different subject. Possible values range from 1 (no disorder at all) to 6 (disturbed sleep each night).

Usage

```
sleep
```

Format

A march dataset.

`sleep_df` *Sleep disorders (data frame format)*

Description

This dataset contains 1000 sequences of 5 to 7 successive observation of the level of sleep disorders. Each sequence corresponds to a different subject. Possible values range from 1 (no disorder at all) to 6 (disturbed sleep each night).

Usage

```
sleep_df
```

Format

A data frame.

Index