Package ‘genMCMCDiag’

July 12, 2024

Title Generalized Convergence Diagnostics for Difficult MCMC Algorithms

Version 0.2.2

Description Trace plots and convergence diagnostics for Markov Chain Monte Carlo (MCMC) algorithms on highly multivariate or unordered spaces. Methods outlined in a forthcoming paper.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

URL `https://github.com/LukeDuttweiler/genMCMCDiag`

BugReports `https://github.com/LukeDuttweiler/genMCMCDiag/issues`

LazyData true

Imports ggplot2 (>= 3.0.0), coda (>= 0.19.0), mcmcse (>= 1.0.0), knitr (>= 1.30), lifecycle

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Luke Duttweiler [aut, cre, cph] (<`https://orcid.org/0000-0002-0467-995X`>)

Maintainer Luke Duttweiler <`lduttweiler@hsph.harvard.edu`>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2024-07-12 19:50:10 UTC

Contents

`bnMCMCResults` *Results from a Bayesian Network MCMC algorithm on simulated data*

Description

Results from a Bayesian Network Metropolis-Hastings algorithm run on simulated data. Included for examples.

Usage

```
bnMCMCResults
```

Format

`bnMCMCResults`**:**

A list with 5 elements, each representing a different MCMC chain. Each element is a list of data.frames describing a partition arrangement of a Bayesian Network.

Source

Luke Duttweiler

`dpmmDistance` *DPMM Distance*

Description

For an MCMC draw from a DPMM D_x, let Z_x be the vector of Z-scores of the observations based on that observation's current group, and let A_x be the 0,1 adjacency matrix where

$$[A_x]_{ij} = 1$$

if observations i and j are in the same group in draw D_x (so the diagonal is always 1s). Then we define the DPMM distance between D_x and D_y as:

$$d(D_x, D_y) = |Z_x - Z_y|^T (|A_x - A_y| + I)|Z_x - Z_y|.$$

Usage

```
dpmmDistance(x, y)
```

Arguments

`x`	List with elements 'Zscore' and 'Adj'
`y`	List with elements 'Zscore' and 'Adj', both of same dimensions as in x.

Value

Numeric, DPMM distance between x and y.

Note

For speed, no error handling if x and y do not have the same dimensions. The function will break if 'Zscore' or 'Adj' doesn't exist though.

`estimateTsTime` *Uses generic formulas and rough time estimate to estimate time it will take to evaluate the TS algorithm on a set of unique draws with the tsTransform function.*

Description

Uses generic formulas and rough time estimate to estimate time it will take to evaluate the TS algorithm on a set of unique draws with the tsTransform function.

Usage

```
estimateTsTime(distance, draw1, draw2, N)
```

Arguments

`distance`	Function with two parameters x,y. Used to calculate distance between draw1 and draw2
`draw1`	Object that works as an argument for distance()
`draw2`	Different object that works as an argument for distance()
`N`	Number of unique draws for which the user is interested in evaluating the time to completion for the TS algorithm

Value

Data.frame with 1 row and 2 columns. Entry one gives the standard completion time, entry two gives the completion time if the fuzzy approximation is used.

`eucDist` *Euclidean Distance*

Description

Simple function to return the Euclidean distance between two objects. Acts elementwise.

Usage

```
eucDist(x, y)
```

Arguments

`x`	Numeric vector or matrix.
`y`	Numeric vector or matrix of same dimensions as x.

Value

Numeric, elementwise Euclidean distance between x and y.

Note

For speed, no error handling if x and y do not have the same dimensions, take care!

Examples

```
eucDist(c(0,0), c(1,1))
```

`fitNN` *Helpful mini function to fit the nearest neighbor (NN) algorithm given a set and defined distance*

Description

Helpful mini function to fit the nearest neighbor (NN) algorithm given a set and defined distance

Usage

```
fitNN(uniqueDraws, uniqueLabels, distance, minDist)
```

Arguments

`uniqueDraws`	List of unique values that make up the set on which we are using the NN algorithm
`uniqueLabels`	List of unique labels associated 1-1 with the unique values
`distance`	Function with arguments x,y that returns a distance defined on the given values
`minDist`	Minimum possible distance between two points that aren't equivalent. May be ignored, but if possible to specify, may speed up the algorithm.

Value

List. tsSolution gives the ordered labels, tsValues gives the ordered values, tsDiffs is a vector of distances between consecutive values in tsValues

`genDiagnostic` *Generate Generalized Diagnostics for Markov Chain Monte Carlo Draws*

Description

This function generates generalized diagnostics for Markov Chain Monte Carlo (MCMC) draws, transforming the draws if specified, and evaluating selected diagnostics.

Usage

```
genDiagnostic(
  mhDraws,
  method = c("standard", "ts", "lanfear", "likelihood"),
  diagnostics = c("traceplot", "ess", "gelmanRubin"),
  distance = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`mhDraws`	A list of MCMC draws, where each element is an ordered list or numeric vector representing the output of a single MCMC chain.
`method`	Method for transforming the MCMC draws. Options include 'standard', 'ts', 'lanfear', or a custom transformation function. See details.
`diagnostics`	A character vector or list of diagnostic functions to be evaluated. Options include 'traceplot', 'ess', 'gelmanRubin', or custom functions. See details.
`distance`	Function for evaluating distance between MCMC draws if required by 'method'. This should be a pairwise distance function that operates on elements of the chains from mhDraws. Note that the lanfear and ts methods ALWAYS require a distance function.
`verbose`	If TRUE, informative messages are displayed.
`...`	Arguments passed on to `tsTransform`, `lanfearTransform`

`minDist` Numeric. Value which specifies the minimum possible distance for two draws which are not equal. See tsTransform details.

`fuzzy` Logical. If TRUE computes an approximate version of the TS algorithm. See tsTransform details.

`fuzzyDist` Numeric. Parameter for approximate version of ts algorithm. See tsTransform details.

`reference` Argument for method = 'lanfear'. Reference point for lanfearTransform (with exact same structure as each MCMC draw) for draw comparison. If left NULL a random point is selected from the given draws. See lanfearTransform details.

Details

Built-in transformation methods can be called with the appropriate character string in the 'method' argument. For details on a particular method use ?lanfearTransform or ?tsTransform. Custom transform functions may be added as well. A custom function must be written to accept a list of mcmcChain type objects, and output a list of dataframes with columns val (the transformed draw) and t (the MCMC chain order). Each element in the list is the transformed MCMC chain corresponding to the input.

Built-in diagnostics can be called with the appropriate character string in the 'diagnostics' argument. Additional custom diagnostic functions may be written. These functions should act on a list of data.frames output from a transform function and should output as a relatively small data.frame where the name of diagnostic is the first row.name.

Value

An object of class 'mcmcDiag', containing evaluated diagnostics, transformed draws, and function call details.

Examples

```
#Example using standard Traceplot
tstS <- genDiagnostic(uniMCMCResults)
```

```
tstS

#Example using 'lanfear' traceplot
tstL <- genDiagnostic(uniMCMCResults, method = 'lanfear', distance = eucDist,
                      reference = 0)
tstL


#Example using bayesian network sample data, with 'lanfear' method
tstBN <- genDiagnostic(bnMCMCResults, method = 'lanfear', distance = partitionDist)
tstBN
```

---

hammingDist *Hamming Distance*

Description

Simple function to return the Hamming distance between two objects. Acts elementwise.

Usage

```
hammingDist(x, y)
```

Arguments

x	Binary vector or matrix
y	Binary vector or matrix of same dimensions as x.

Value

Numeric, elementwise Hamming distance between x and y.

Note

For speed, no error handling if x and y do not have the same dimensions. Also, does not test to make sure x,y are binary, take care!

Examples

```
x <- matrix(c(1,0,
              0,0), nrow = 2, byrow = TRUE)
y <- diag(1,2)
hammingDist(x, y)
```

lanfearTransform *Transforms a list of MCMC chains into a list of data.frames using the Lanfear transformation*

Description

Transforms a list of MCMC chains into a list of data.frames using the Lanfear transformation

Usage

```
lanfearTransform(mhDraws, distance, reference = NULL, ...)
```

Arguments

`mhDraws`	List. Each element is a single chain from an MCMC algorithm. Each element should be a numeric vector (for univariate draws), or a list.
`distance`	Distance function defined on the space of MCMC draws. Should operate pairwise on the elements of the given chains. See details.
`reference`	Argument for method = 'lanfear'. Reference point for lanfearTransform (with exact same structure as each MCMC draw) for draw comparison. If left NULL a random point is selected from the given draws. See lanfearTransform details.
`...`	Catches extra arguments. Not used.

Details

The Lanfear transformation works by specifying a reference point and then comparing each MCMC draw back to that reference point using a distance function. The function returns this distance value as the Lanfear transformation of each draw.

Value

List of data.frames with columns 'val' which is the Lanfear transformation of each MCMC draw, and 't' which gives the within-chain ordering of the MCMC draws. Each data.frame is a separate chain.

listLabels *Function to assign character labels to all unique objects in a list*

Description

Function to assign character labels to all unique objects in a list

Usage

```
listLabels(lst)
```

Arguments

`lst`	A list of objects. Each object in the list should have the same general structure

Value

A character vector of labels. Objects in lst that are identical will be assigned the same label.

`partitionDist` *Partition Distance*

Description

Function to return the 'Partition' distance between two objects. Used for Bayesian Networks with the 'partition-MCMC' algorithm.

Usage

```
partitionDist(x, y)
```

Arguments

`x`	Data.frame with columns node and partition
`y`	Data.frame with columns node and partition. Same nrows as x.

Value

Numeric, Partition distance between x and y.

Note

For speed, no error handling if x and y do not have the same dimensions. Also, does not test to make sure x,y are data.frames of integers, take care!

Examples

```
x <- bnMCMCResults[[1]][[1]]
y <- bnMCMCResults[[1]][[100]]
partitionDist(x, y)
```

`print.mcmcDiag` *Print method for mcmcDiag objects*

Description

Print method for mcmcDiag objects

Usage

```
## S3 method for class 'mcmcDiag'
print(x, ...)
```

Arguments

`x`	Object of class mcmcDiag
`...`	Kept for consistency with print. Does nothing.

Value

Invisible NULL, prints to console

Examples

```
print(genDiagnostic(uniMCMCResults))
```

`sess` *Calculate the effective sample size, per chain and in total, of draws from an MCMC algorithm*

Description

Calculate the effective sample size, per chain and in total, of draws from an MCMC algorithm

Usage

```
sess(mhDraws, ...)
```

Arguments

`mhDraws`	List of data.frames. Each data.frame represents a single chain. Data.frame columns for which ESS is calculated should be named val.1, ..., val.k
`...`	Catches unnecessary additional arguments

Value

Data.frame with 1 Row and (# Chains + 1) Columns. Each entry gives the estimated ESS for the chain or sum of chains.

`sgelmanRubin` *Calculate the Gelman-Rubin diagnostic of draws from an MCMC algorithm*

Description

Calculate the Gelman-Rubin diagnostic of draws from an MCMC algorithm

Usage

```
sgelmanRubin(mhDraws, ...)
```

Arguments

`mhDraws`	List of data.frames with two columns. Each data.frame represents a single chain. Column names should be val.1 (for values) and t (for chain iteration).
`...`	Catches unnecessary additional arguments

Value

Data.frame with 1 row and 2 columns. First entry gives estimated GR statistic, second gives upper 95% limit for GR statistic.

`standardTransform` *Transforms a list of MCMC chains into a list of dataframes with no modifications to values*

Description

Transforms a list of MCMC chains into a list of dataframes with no modifications to values

Usage

```
standardTransform(mhDraws, ...)
```

Arguments

`mhDraws`	An list of numeric vectors
`...`	Not used.

Value

A list of data.frames with rows that represent MCMC draws.Each separate data.frame is a different chain. Data.frames have columns 'val' for the numeric draws, and 't' for the draw. Currently, using the standard transformation on anything other than univariate draws is not supported.

`straceplot` *Generate a traceplot of draws from a multi-chain MCMC*

Description

Generate a traceplot of draws from a multi-chain MCMC

Usage

```
straceplot(mhDraws, method = NULL, ...)
```

Arguments

`mhDraws`	List of data.frames with two columns. Each data.frame represents a single chain. Column names should be val.1 (for values) and t (for chain iteration).
`method`	Character string - Name of method used to generate traceplot. Is used to generate the title of the traceplot.
`...`	Catches unused arguments

Value

ggplot2 plot object showing traceplot

`tsTransform` *Transforms a list of MCMC chains into a list of data.frames using the TS transformation*

Description

Transforms a list of MCMC chains into a list of data.frames using the TS transformation

Usage

```
tsTransform(
  mhDraws,
  distance,
  minDist = 0,
  fuzzy = FALSE,
  fuzzyDist = 1,
  verbose = FALSE,
  ...
)
```

Arguments

`mhDraws`	List. Each element is a single chain from an MCMC algorithm. Each element should be a numeric vector (for univariate draws), or a list.
`distance`	Distance function defined on the space of MCMC draws. Should operate pair-wise on the elements of the given chains. See details.
`minDist`	Numeric. Value which specifies the minimum possible distance for two draws which are not equal. See tsTransform details.
`fuzzy`	Logical. If TRUE computes an approximate version of the TS algorithm. See tsTransform details.
`fuzzyDist`	Numeric. Parameter for approximate version of ts algorithm. See tsTransform details.
`verbose`	Logical. If TRUE, function prints out information about approximate computation time
`...`	Catches extra arguments. Not used.

Details

The TS transformation sets up a traveling salesman algorithm by calculating the pair-wise distances between each unique draw from the mhDraws and solving the resulting TS problem with the nearest neighbor (NN) algorithm.

minDist can be used to speed up the algorithm if it is known that when x != y then distance(x, y) >= minDist. Otherwise this should be ignored.

The fuzzy approximation of the algorithm works by splitting the unique draws into smaller sets each containing at most 1% of all unique draws, and fitting the NN algorithm within each set, and then on the resulting 'end points' of each set. The sets are created by randomly selecting a representative draw and then putting the 'closest' draws with distance less than fuzzyDist into that set, until the set contains 1% of all unique draws. The fuzzy approximation can GREATLY reduce computation time, unless the fuzzyDistance specified is too small.

Value

List of data.frames with columns 'val' which is the TS transformation of each MCMC draw, and 't' which gives the within-chain ordering of the MCMC draws. Each data.frame is a separate chain.

`uniMCMCResults` *Results from a univariate MCMC algorithm on a simulated posterior*

Description

Results from a univariate Metropolis-Hastings algorithm run on a tri-modal posterior. Although the standard traceplot and Gelman-Rubin diagnostic show good mixing, the results are actually mixing poorly. Included for examples.

Usage

```
uniMCMCResults
```

Format

`uniMCMCResults`**:**
A list with 7 elements, each representing a different MCMC chain. Each element is a numeric vector of length 2000

Source

Luke Duttweiler

Index