Package ‘MinCompSpinPackage’

January 26, 2022

Type Package

Title Uncover and model community structures in binary data by doing an exhaustive search for the best Minimally Complex Model (MCM)

Version 1.0

Date 2021-11-15

Description An interface for the original C++ based program MinCompSpin.
This program allows to uncover AND model community structures in binary data,
while taking into account possible high order patterns of data in the
detection of the communities (i.e., possible high-order correlations between the variables).
The idea of the algorithm is based on performing statistical inference
with a family of spin models (maximum entropy models for binary data)
that have minimal information theoretic complexity.
These models are called Minimally Complex Models (MCM).
The selected model can be used as a generative model for data. This package
is good for use on small systems, typically with <~15 variables
per binary data-point. Note that for ~15 variables, it may take several
days to perform the exhaustive search since the number of possible models
grows exponentially with the number of variables.

License GPL (>= 3)

Encoding UTF-8

SystemRequirements C++11

Depends R (>= 4.1.2)

Imports Rcpp (>= 1.0.7)

LinkingTo Rcpp

RoxygenNote 7.1.2

R topics documented:

`build_Kset` *Build K_set*

Description

Build Kset for the states written in the basis of the m-chosen independent operator on which the SC model is based. Changes the basis of the data-set from its original basis to the basis provided as an argument in basis. Print Kset to terminal by setting print_bool to TRUE.

Usage

```
build_Kset(Nset, Basis, n, print_bool = FALSE)
```

Arguments

`Nset`	A list containing 2 vectors which correspond to a index based mapping of the states in the data set to their occurrence.
`Basis`	The basis on which the model is based.
`n`	Number of binary (spin) variables of the data set.
`print_bool`	print relevant information. Default: FALSE.

Value

Kset Mapping of frequency of occurrence of each state of the data-set in the new basis.

`check_partition` *check_partition*

Description

Check if *Partition* is an actual partition of r basis elements, i.e., that no basis element appears in more than 1 part of the partition. i.e., that each basis element only appears in a single part of the partition.

Usage

```
check_partition(Partition, n)
```

Arguments

`Partition`	Partition of MCM.
`n`	The amount of binary (spin) variables.

Value

Bool_results Returns FALSE if there is an overlap between the parts.

`Complexity_MCM` *Complexity of an MCM based on the geometric and parametric complexity.*

Description

Complexity of and MCM defined by the addition of the total geometric complexity and total (parametric) first order complexity. Compute separately: – the first order complexity –> stored in C_param – and the geometric complexity –> stored in C_geom

Usage

```
Complexity_MCM(Partition, N, n, C_param, C_geom)
```

Arguments

`Partition`	Partition of MCM.
`N`	The size of the dataset
`n`	The amount of spin variables.
`C_param`	The (parametric) first order complexity.
`C_geom`	The geometric complexity.

Value

Complexity A double containing the complexity of the MCM.

`Create_MCM` *Define an MCM*

Description

Define an MCM. Input is an integer representation of all MCM parts. For example, MCM_Choice = c(384, 64, 32, 16, 8, 4, 2, 1) defines an MCM with 8 independent parts, based on n=9 spins. You can check that the model (i.e., list of parts) that you have provided properly defines an MCM by calling the function check_partition(Partition, n).

Usage

```
Create_MCM(MCM_table)
```

Arguments

`MCM_table`	List containing the integer representation of the MCM parts.

Value

MCM_partition The defined MCM.

`GeomComplexity_SubCM` *GeomComplexity_SubCM*

Description

Geometric Complexity of a Sub-Complete Model based on m basis operators. for 1 <= m <= n . Rem C(m=1) = log(pi)

Usage

```
GeomComplexity_SubCM(m)
```

Arguments

`m`	The number of basis elements/operators.

Value

GeomComplexity The complexity.

`LogE_MCM` *Compute the log-evidence of of an MCM.*

Description

Compute the log-evidence of of an MCM.

Usage

```
LogE_MCM(Kset, Partition, N, n, print_bool = FALSE)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`Partition`	Partition of MCM.
`N`	The size of the data-set.
`n`	The amount of binary (spin) variables.
`print_bool`	Boolean to print related information to terminal. Default: FALSE.

Value

LogE The log-evidence of an MCM as a double.

`LogE_SubCM` *Compute the log-evidence of the sub-complete part (of an MCM).*

Description

Compute the log-evidence of the sub-complete part (of an MCM) defined by Ai. This function could be also used directly by the user to compute the log-likelihood of a sub-complete model

Usage

```
LogE_SubCM(Kset, Ai, N, n, print_bool = FALSE)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`Ai`	The binary representation of the sub-complete part (of an MCM).
`N`	The size of the data-set.
`n`	The amount of binary (spin) variables.
`print_bool`	Boolean to print related information to terminal. Default: FALSE.

Value

LogE The log-evidence of the sub-complete part (of an MCM) as a double.

`LogL_CM` *Compute the log-likelihood of a complete model on Kset.*

Description

Compute the log-likelihood of a complete model on Kset: This function is mainly used for call by 'LogL_SC_PartMCM', but can also be used to compute the log-likelihood of a complete model

Usage

```
LogL_CM(Kset, N)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`N`	The size of the data-set.

Value

LogL The log-likelihood of a complete model on Kset as a double.

`LogL_MCM` *Compute log-likelihood of an MCM.*

Description

Compute the log-likelihood of of an MCM. This function could be also used directly by the user to compute the log-likelihood of a model.

Usage

```
LogL_MCM(Kset, Partition, N, n, print_bool = FALSE)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`Partition`	Partition of MCM.
`N`	The size of the data-set.
`n`	Number of binary (spin) variables of the data set.
`print_bool`	Boolean to print related information to terminal. Default: FALSE.

Value

LogL The log-likelihood of a complete model on Kset as a double.

`LogL_SubCM` *Compute log-likelihood of the sub-complete part (SC-part) of an MCM.*

Description

Compute the log-likelihood of the sub-complete part (of an MCM) defined by Ai. This function could be also used directly by the user to compute the log-likelihood of a sub-complete model

Usage

```
LogL_SubCM(Kset, Ai, N, n, print_bool = FALSE)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`Ai`	The binary representation of the SC-part.
`N`	The size of the data-set.
`n`	Number of binary (spin) variables of the data set.
`print_bool`	Boolean to print related information to terminal. Default: FALSE.

Value

LogL The log-likelihood of a sub-complete part (of an MCM) defined by Ai.

`MCM_AllRank_SmallerThan_r_nonOrdered`
MCM_AllRank_SmallerThan_r_Ordered

Description

Compare all the MCMs based on any subset of k elements of the of r first elements of the basis used to build Kset for all k=1 to r, where r <= basis.size() By default: - r=n - the function doesn't print the logE-values for all the tested MCMs. To activate –> print_bool = true

Usage

```
MCM_AllRank_SmallerThan_r_nonOrdered(
  Kset,
  N,
  LogE_best,
  r,
  n,
  OUTPUT_directory,
  print_bool
)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`N`	The size of the dataset
`LogE_best`	The double which will contain the maximized Log-evidence.
`r`	The rank r which is the r first elements of the basis.
`n`	The amount of binary (spin) variables.
`OUTPUT_directory`	The name of the output directory (and path to it) in which to save the output data.
`print_bool`	Boolean to print related information to terminal. Default: FALSE.

Value

Partition The best MCM.

MCM_AllRank_SmallerThan_r_Ordered

MCM_AllRank_SmallerThan_r_Ordered

Description

Compare all the MCM based on the k first elements of the basis used to build Kset for all k=1 to r, where r <= basis.size(). By default: - r=n - the function doesn't print the logE-values for all the tested MCMs. To activate –> print_bool = true

Usage

```
MCM_AllRank_SmallerThan_r_Ordered(
  Kset,
  N,
  LogE_best,
  r,
  n,
  OUTPUT_directory,
  print_bool
)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`N`	The size of the data-set.
`LogE_best`	The double which will contain the maximized Log-evidence.
`r`	The rank r which is the r first elements of the basis.
`n`	The amount of binary (spin) variables.
`OUTPUT_directory`	The name of the output directory (and path to it) in which to save the output data.
`print_bool`	Boolean to print related information to terminal. Default: FALSE.

Value

Partition The best MCM.

`MCM_GivenRank_r` *MCM_GivenRank_r*

Description

Compare all the MCM of rank r, based on the r first elements of the basis used to build Kset:

Usage

```
MCM_GivenRank_r(Kset, N, LogE_best, r, n, OUTPUT_directory, print_bool)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`N`	The size of the data-set.
`LogE_best`	The double which will contain the maximized Log-evidence.
`r`	The rank r which is the r first elements of the basis.
`n`	The amount of binary (spin) variables.
`OUTPUT_directory`	The name of the output directory (and path to it) in which to save the output data.
`print_bool`	Boolean to print related information to terminal. Default: FALSE.

Value

Partition The best MCM.

`MCM_search` *Perform the exhaustive search for the best MCM at once.*

Description

Performs the community detection on the data given in datafilename with the given basis Basis_li and returns the best fitting MCM for the data. Optional: set the comparison variable to TRUE to test what happens when the model is compared to 3 different cases. For general use set to FAlSE. Output is stored in the (newly created) OUTPUT directory in the same working directory.

Usage

```
MCM_search(n, datafilename, Basis_li, OUTPUT_directory, comparison)
```

Arguments

`n`	Number of binary (spin) variables of the data set.
`datafilename`	The name of input file including path containing the binary data.
`Basis_li`	The basis on which the model is based.
`OUTPUT_directory`	The name of the output directory (and path to it) in which to save the output data.
`comparison`	Boolean indicating whether to run 2 other versions of the exhaustive search. For comparison purposes. Default: FALSE.

`Original_Basis` *Original Basis*

Description

Return the original basis, i.e., s1, s2, ..., sn

Usage

```
Original_Basis(n)
```

Arguments

`n`	Number of binary (spin) variables of the data set.

Value

Basis_li List of the original basis i.e. s1, s2, ..., sn.

`ParamComplexity_SubCM` *Parametric Complexity of a sub-complete model*

Description

Parametric Complexity of a Sub-Complete Model based on m basis operators. for 1 <= m <= n . Rem C(m=1) = log(pi).

Usage

```
ParamComplexity_SubCM(m, N)
```

Arguments

`m`	The number of basis elements/operators.
`N`	The size of the data-set.

Value

ParamComplexity The complexity.

`PrintInfo_All_Indep_Models` *PRINT INFO ON SUCCESSIVE INDEPENDENT MODELS IN THE NEW BASIS.*

Description

The function prints the integer and binary representation of the part , as well as the the log-likelihood (LogL) and the log-evidence (LogE).

Usage

```
PrintInfo_All_Indep_Models(Kset, N, n)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`N`	The size of the data-set.
`n`	The amount of binary (spin) variables.

`PrintInfo_All_SubComplete_Models` *PrintInfo_All_Indep_Models*

Description

PRINT INFO ON SUCCESSIVE SUB_COMPLETE MODELS IN THE NEW BASIS.

Usage

```
PrintInfo_All_SubComplete_Models(Kset, N, n)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`N`	The size of the data-set.
`n`	The amount of binary (spin) variables.

`PrintTerminal_MCM_Info` *PRINT INFO on each PART of an MCM (= a partition)*

Description

The function prints the number of Independent Components of the MCM (i.e., the number of partitions – or communities – of the MCM), as well as the log-likelihood (LogL), the parameter complexity (C_param), the geometric complexity (C_geom), the total complexity (C_tot), the Minimum Description Length (MDL) and the log-evidence (LogE).

Usage

```
PrintTerminal_MCM_Info(Kset, N, n, MCM_Partition)
```

Arguments

`Kset`	Mapping of frequency of occurrence of each state of the data-set in the new basis.
`N`	The size of the data-set.
`n`	The amount of binary (spin) variables.
`MCM_Partition`	Partition of MCM.

`PrintTerm_Basis` *Print Basis*

Description

Print the basis info in the terminal.

Usage

```
PrintTerm_Basis(Basis_li, n)
```

Arguments

`Basis_li`	A list containing the basis.
`n`	Number of binary (spin) variables of the data set.

`Read_BasisOp_BinaryRepresentation` *Read basis operators in binary representation from input file.*

Description

Reads the Basis operators in binary representation from the given input file. The operators should be written in one single column and as binary representation of the spin involved in the Operator. Returns a list of the basis with the integer value of that binary representation

Usage

```
Read_BasisOp_BinaryRepresentation(Basis_binary_filename, n)
```

Arguments

`Basis_binary_filename`	The name of input file including path.
`n`	Number of binary (spin) variables of the data set.

Value

Basis_li List of the basis with the integer value of that binary representation.

`Read_BasisOp_IntegerRepresentation` *Read basis operators in integer representation from input file.*

Description

Reads the Basis operators in integer representation from the given input file. The operators should be written in one single column and as the integer value of that binary representation. Returns a list of the given input file.

Usage

```
Read_BasisOp_IntegerRepresentation(Basis_integer_filename)
```

Arguments

`Basis_integer_filename`	The name of input file including path.

Value

Basis_li List of the given input file.

`read_datafile` *Read data and store in Nset.*

Description

Reads the binary data set (with location and name provided in argument filename) as binary integers and returns a mapping of each observed state to the number of times they occur in the data set. Note that each state of the system is encoded as an n-bit integer.

Usage

```
read_datafile(N, n, filename)
```

Arguments

`N`	Integer that will contain the data set size.
`n`	Number of binary (spin) variables of the data set.
`filename`	The string containing the path to the data file and the name.

Value

Nset The first row contains the observed state and the second row contains their occurrence (matching column-wise).

`Read_MCMParts_BinaryRepresentation`
Read_MCMParts_BinaryRepresentation

Description

Define MCM from a file in which the operators are written as the binary representation of the interactions.

Usage

```
Read_MCMParts_BinaryRepresentation(MCM_binary_filename, n)
```

Arguments

`MCM_binary_filename`	The file name (incl. the path to it from the current directory).
`n`	The amount of binary (spin) variables.

Value

MCM_partition The defined MCM.

Index