Package ‘stablespec’

October 14, 2022

**Type** Package

**Title** Stable Specification Search in Structural Equation Models

**Version** 0.3.0

**Date** 2017-04-04

**Description** An exploratory and heuristic approach for specification search in Structural Equation Modeling. The basic idea is to subsample the original data and then search for optimal models on each subset. Optimality is defined through two objectives: model fit and parsimony. As these objectives are conflicting, we apply a multi-objective optimization method, specifically NSGA-II, to obtain optimal models for the whole range of model complexities. From these optimal models, we consider only the relevant model specifications (structures), i.e., those that are both stable (occur frequently) and parsimonious and use those to infer a causal model.

**URL** [https://github.com/rahmarid/stablespec](https://github.com/rahmarid/stablespec)

**BugReports** [https://github.com/rahmarid/stablespec/issues](https://github.com/rahmarid/stablespec/issues)

**Depends** R (>= 3.1.0)

**License** MIT + file LICENSE

**LazyData** TRUE

**Imports** ggm, matrixcalc, sem, nsga2R, graph, Rgraphviz, methods, polycor, foreach

**RoxygenNote** 6.0.1

**Suggests** testthat

**NeedsCompilation** no

**Author** Ridho Rahmadi [aut, cre], Perry Groot [aut, ths], Tom Heskes [aut, ths], Christoph Stich [ctb]

**Maintainer** Ridho Rahmadi <r.rahmadi@cs.ru.nl>

**Repository** CRAN

**Date/Publication** 2017-04-05 03:27:52 UTC
**R topics documented:**

- `crossdata6V` ........................................... 2
- `dataReshape` ........................................... 2
- `getModelFitness` .................................... 3
- `longiData4V3T` ....................................... 4
- `modelPop` ............................................. 4
- `plotStability` ........................................ 5
- `repairCyclicModel` ................................... 7
- `stableSpec` ........................................... 8

**Index**

```
crossdata6V  Artificial cross-sectional data.
```

**Description**

A data set of 500 instances, generated from a network consisting of six continuous variables.

**Usage**

`crossdata6V`

**Format**

A data frame with six continuous variables: `x1, ..., x6`.

```
dataReshape  Reshape longitudinal data
```

**Description**

Reshape longitudinal data with `t` time slices into a longitudinal data with two time slices.

**Usage**

`dataReshape(theData = NULL, numTime = NULL)`

**Arguments**

- `theData` a data frame containing longitudinal data to which the model will be fit.
- `numTime` number of time slices.
getModelFitness

Value

A data frame representing longitudinal data with two time slices, such that the first \( n \) data points contain the relations that occur in the first two time slices \( t_0 \) and \( t_1 \). The next \( n \) data points contain the relations that occur in time slices \( t_1 \) and \( t_2 \). The \( i \)-th subset of \( n \) data points contain the relations in time slices \( t_{i-1} \) and \( t_i \). The reshaped data can be used as data input for function `stableSpec` when computing longitudinal data.

Author(s)

Ridho Rahmadi <r.rahmadi@cs.ru.nl>

Examples

```r
the_data <- longiData4V3T
num_time <- 3
reshaped_the_data <- dataReshape(the_data, num_time)
```

getModelFitness

Scoring the given SEM models.

Description

Compute the model chi-square and model complexity of the given SEM models.

Usage

```r
gemodelFitness(theData = NULL, allModelString = NULL, numTime = NULL, longitudinal = NULL, co = NULL, mixture = NULL)
```

Arguments

- **theData**: a data frame containing the data to which the model is to be fit. If parameter `longitudinal` is `TRUE`, the data frame should be reshaped such that the first \( n \) data points contain the relations that occur in the first two time slices \( t_0 \) and \( t_1 \). The next \( n \) data points contain the relations that occur in time slices \( t_1 \) and \( t_2 \). The \( i \)-th subset of \( n \) data points contain the relations in time slices \( t_{i-1} \) and \( t_i \). One can use function `dataReshape` to reshape longitudinal data.

- **allModelString**: \( m \) by \( n \) matrix of binary vectors representing models, where \( m \) is the number of models, and \( n \) is the length of the binary vector.

- **numTime**: number of time slices. If the data is cross-sectional, this argument must be set to 1.

- **longitudinal**: `TRUE` for longitudinal data, and `FALSE` for cross-sectional data.

- **co**: whether to use "covariance" or "correlation" matrix.

- **mixture**: if the data contains both continuous and categorical (or ordinal) variables, this argument can be set to `TRUE`. This implies the use of polychoric and polyserial correlation in the SEM computation. Note that, the categorical variables should be represented as factor or logical.
Value

A **matrix** of models including their fitness: chi-square and model complexity.

Author(s)

Ridho Rahmadi <r.rahmadi@cs.ru.nl>

Examples

```r
the_data <- crossdata6V
# assumed that variable 5 does not cause variables 1, 2, and 3
models <- modelPop(nPop=5, numVar=6, longitudinal=FALSE,
                    consMatrix = matrix(c(5, 1, 5, 2, 5, 3), 3, 2, byrow=TRUE))

model_fitness <- getModelFitness(theData=the_data,
                                   allModelString=models, numTime=1, longitudinal=FALSE,
                                   co="covariance", mixture = FALSE)
```

Description

A data set of 400 instances, that is generated from a network consisting of four continuous variables and three time slices $t_0, \ldots, t_2$.

Usage

```r
longiData4V3T
```

Format

A data frame with twelve continuous variables: $x_1, \ldots, x_4$ are for time slice $t_0$, $x_5, \ldots, x_8$ are for time slice $t_1$, and $x_9, \ldots, x_{12}$ are for time slice $t_2$.

Description

Generating recursive (acyclic) SEM models represented by binary vectors.

Usage

```r
modelPop(nPop = NULL, numVar = NULL, longitudinal = NULL,
         consMatrix = NULL)
```
Arguments
nPop number of models to generate or population size.
numVar number of variables.
longitudinal TRUE for longitudinal data, and FALSE for cross-sectional data.
consMatrix m by 2 binary matrix representing constraint/prior knowledge, where m is the number of constraint. For example, known that variables 2 and 3 do not cause variable 1, then constraint <- matrix(c(2, 1, 3, 1), 2, 2, byrow=TRUE)) will be the constraint matrix.

Details
This function generates nPop random SEM models which are represented by binary vectors; 1 means there is a causal path from, e.g., variable A to B and 0 otherwise. In addition, the generated models have passed the cyclic test to ensure they are all acyclic. The function also includes minPop models which representing models from each model complexity, i.e., minPop = numVar(numVar-1)/2+1, if longitudinal = FALSE, or minPop = (numVar(numVar-1)/2+1)+numVar^2, otherwise. If nPop <= minPop then this function will generate minPop models.

Value
nPop or minPop by m matrix, where m is the length of the binary vector depending of the given number of variables and also whether longitudinal or cross-sectional model.

Author(s)
Ridho Rahmadi <r.rahmadi@cs.ru.nl>

Examples
# assuming a prior knowledge that variable 1 does not cause variable 2
models <- modelPop(nPop=25, numVar=6,
 longitudinal=FALSE, consMatrix = matrix(c(1, 2), 1, 2))
models

plotStability

Plot of edge and causal path stability.

Description
Plot each of the stability of causal path and edge including the threshold of stability and model complexity.

Usage
plotStability(listOfFronts = NULL, threshold = NULL, stableCausal = NULL, stableCausal_l1 = NULL, stableEdge = NULL, longitudinal = NULL)
plotStability

Arguments

listOfFronts  list of models including their fitness and subset index.
threshold  threshold of stability selection. The default is 0.6.
stableCausal  list of causal path stability for the whole range of model complexities.
stableCausal_l1  list of causal path stability of length 1 for the whole range of model complexities.
stableEdge  list of edge stability for the whole range of model complexities.
longitudinal  TRUE for longitudinal data, and FALSE cross-sectional data.

Value

Plot of causal path and edge stability for every pair of variables, including plots of all edge stabilities and all causal path stabilities.

Author(s)

Ridho Rahmadi <r.rahmadi@cs.ru.nl>

Examples

the_data <- crossdata6V
numSubset <- 1
num_iteration <- 5
num_pop <- 10
mut_rate <- 0.075
cross_rate <- 0.85
longi <- FALSE
num_time <- 1
the_co <- "covariance"
#assumed that variable 5 does not cause variables 1, 2, and 3
cons_matrix <- matrix(c(5, 1, 5, 2, 5, 3), 3, 2, byrow=TRUE)
th <- 0.1
to_plot <- FALSE

result <- stableSpec(theData=the_data, nSubset=numSubset, iteration=num_iteration,
num_pop=num_pop, mut_rate=mut_rate, cross_rate=cross_rate,
longitudinal=longi, num_time=num_time,
co=the_co, consMatrix=cons_matrix, threshold=th, toPlot=to_plot)

plotStability(listOfFronts=result$listOfFronts, threshold=th,
stableCausal=result$causalStab,
stableCausal_l1=result$causalStab_l1,
stableEdge=result$edgeStab,
longitudinal=longi)
repairCyclicModel

Reparing a SEM model that is cyclic.

Description

Repairing a SEM model that is cyclic.

Usage

repairCyclicModel(stringModel = NULL, numVar = NULL, longitudinal = NULL)

Arguments

stringModel binary vector with length n^2+(n(n-1)) if longitudinal = TRUE, or n(n-1) if FALSE, where n is the number of variables (numVar).
numVar number of variables.
longitudinal TRUE for longitudinal data, and FALSE for cross-sectional data.

Details

The main idea of this function is to seek cyclic(s) with any possible length from a given model, and then to cut the cyclic, so as to make the model acyclic. Moreover, this function is used in stableSpec to ensure no cyclic model in the computation.

Value

a binary vector with the same length of input, representing a repaired model (acyclic).

Author(s)

Ridho Rahmadi <r.rahmadi@cs.ru.nl>

Examples

num_vars <- 6
longi_a <- FALSE
longi_b <- TRUE

# Assume that the generated model below is cyclic
# a cross-sectional model
model_a <- round(runif(num_vars * num_vars))
# a longitudinal model
model_b <- c(round(runif(num_vars * num_vars)),
round(runif(num_vars * (num_vars-1))))

repaired_model_a <- repairCyclicModel(stringModel=model_a, numVar=num_vars,
longitudinal=longi_a)
repaired_model_b <- repairCyclicModel(stringModel=model_b, numVar=num_vars, longitudinal=longi_b)

repaired_model_a
repaired_model_b

---

**stableSpec**

*Stable specifications of constrained structural equation models.*

**Description**

Search stable specifications (structures) of constrained structural equation models.

**Usage**

```r
stableSpec(theData = NULL, nSubset = NULL, iteration = NULL, nPop = NULL, mutRate = NULL, crossRate = NULL, longitudinal = NULL, numTime = NULL, seed = NULL, co = NULL, consMatrix = NULL, threshold = NULL, toPlot = NULL, mixture = NULL, log = NULL)
```

**Arguments**

- `theData` a data frame containing the data to which the model will be fit. If argument `longitudinal` is `TRUE`, the data frame should be reshaped such that the first `n` data points contain the relations that occur in the first two time slices $t_0$ and $t_1$. The next `n` data points contain the relations that occur in time slices $t_1$ and $t_2$. The $i$-th subset of `n` data points contain the relations in time slices $t_{i-1}$ and $t_i$. One can use function `dataReshape` to reshape longitudinal data. Uses the `foreach` package for parallel computation. You need to register a parallel backend before calling `stableSpec` if you want to parallize computation. For details see the `foreach` package.
- `nSubset` number of subsets to draw. In practice, it is suggested to have at least 25 subsets. The default is 10.
- `iteration` number of iterations/generations for NSGA-II.
- `nPop` population size (number of models) in a generation. The default is 50.
- `mutRate` mutation rate. The default is 0.075.
- `crossRate` crossover rate. The default is 0.85.
- `longitudinal` TRUE for longitudinal data, and FALSE for cross-sectional data.
- `numTime` number of time slices. If the data is cross-sectional, this argument must be set to 1.
- `seed` integer vector representing seeds that are used to subsample data. The default is an integer vector with range 100:1000 with length equal to `nSubset`.
- `co` whether to use "covariance" or "correlation" matrix. The default is "covariance".
**stableSpec**

**consMatrix**  
m by 2 binary matrix representing constraint/prior knowledge, where m is the number of constraint. For example, known that variables 2 and 3 do not cause variable 1, then constraint <- matrix(c(2, 1, 3, 1), 2, 2, byrow=TRUE)) will be the constraint matrix. If NULL, then it is assumed that there is no constraint.

**threshold**  
threshold of stability selection. The default is 0.6.

**toPlot**  
if TRUE a plot of inferred causal model is generated, otherwise a graph object is returned. The default is TRUE.

**mixture**  
if the data contains both continuous and categorical (or ordinal) variables, this argument can be set to TRUE. This implies the use of polychoric and polyserial correlation in the SEM computation. Note that, the categorical variables should be represented as factor or logical.

**log**  
an optional logfile to monitor the progress of the algorithm.

**Details**

This function performs exploratory search over recursive (acyclic) SEM models. Models are scored along two objectives: the model fit and the model complexity. Since both objectives are often conflicting we use NSGA-II to search for Pareto optimal models. To handle the instability of small finite data samples, we repeatedly subsample the data and select those substructures that are both stable and parsimonious which are then used to infer a causal model.

**Value**

a list of the following elements:

- **listofFronts** is a list of optimal models for the whole range of model complexity of all subsets.
- **causalStab** is a list of causal path stability for the whole range of model complexity
- **causalStab_l1** is a list of causal path stability of length 1 for the whole range of model complexity
- **edgeStab** is a list of edge stability for the whole range of model complexity
- **relCausalPath** is n by n matrix of relevant causal path, where n is the number of variables. Each positive element i,j represents the stability of causal path from i to j.
- **relCausalPath_l1** is n by n matrix of relevant causal path with length 1, where n is the number of variables. Each positive element i,j represents the stability of causal path from i to j with length 1.
- **relEdge** is n by n matrix of relevant edge, where n is the number of variables. Each positive element i,j represents the stability of edge between i to j.
- If argument toPlot = TRUE, then a visualization of relevant model structures is generated. Otherwise an object of graph is returned. An arc represents a causal path, and an (undirected) edge represents strong association where the direction is undecidable. The graph is annotated with reliability scores, which are the highest selection probability in the top-left region of the edge stability graph.
- **allSeed** is an integer vector representing seeds that are used in subsampling data. This can be used to replicate the result in next computation.
Author(s)

Ridho Rahmadi <r.rahmadi@cs.ru.nl>, Perry Groot, Tom Heskes. Christoph Stich is the contributor for parallel support.

References


Examples

# Cross-sectional data example,
# with an artificial data set of six continuous variables.
# Detail about the data set can be found in the documentation.
# As an example, we only run one subset.
# Note that stableSpec() uses foreach to support
# parallel computation, which could issue a warning
# when running sequentially as the following example. However
# the warning can be just ignored.

the_data <- crossdata6V
numSubset <- 1
num_iteration <- 5
num_pop <- 10
mut_rate <- 0.075
cross_rate <- 0.85
longi <- FALSE
num_time <- 1
the_seed <- NULL
the_co <- "covariance"
#assumed that variable 5 does not cause variables 1, 2, and 3
```r
cons_matrix <- matrix(c(5, 1, 5, 2, 5, 3), 3, 2, byrow=TRUE)
th <- 0.1
to_plot <- FALSE
mix <- FALSE

result <- stableSpec(theData=the_data, nSubset=numSubset,
iteration=num_iteration,
nPop=num_pop, mutRate=mut_rate, crossRate=cross_rate,
longitudinal=longi, numTime=num_time, seed=the_seed,
co=the_co, consMatrix=cons_matrix, threshold=th,
toPlot=to_plot, mixture = mix)
```

Parallel computation is possible by registering parallel backend, e.g., package doParallel. For example, add the following lines on top of the example above.

```
# library(parallel)
# library(doParallel)
# cl <- makeCluster(detectCores())
# registerDoParallel(cl)
#
# Then call stableSpec() as normal.
```

Note that makeCluster() and detectCores() are from package parallel, and registerDoParallel() is from package doParallel. For more detail check the aforementioned packages' documentations.
Index

* datasets
  - crossdata6V, 2
  - longiData4V3T, 4

  crossdata6V, 2
  dataReshape, 2, 3, 8
  getModelFitness, 3
  list, 6, 9
  longiData4V3T, 4
  matrix, 3–5, 9
  modelPop, 4
  plotStability, 5
  repairCyclicModel, 7
  stableSpec, 3, 7, 8