Package: MplusLGM (via r-universe)

March 6, 2025

Type Package

Title Automate Latent Growth Mixture Modelling in 'Mplus'

Version 1.0.0

Description Provide a suite of functions for conducting and automating Latent Growth Modeling (LGM) in 'Mplus', including Growth Curve Model (GCM), Growth-Based Trajectory Model (GBTM) and Latent Class Growth Analysis (LCGA). The package builds upon the capabilities of the 'MplusAutomation' package (Hallquist & Wiley, 2018) to streamline large-scale latent variable analyses. "MplusAutomation: An R Package for Facilitating Large-Scale Latent Variable Analyses in Mplus." Structural Equation Modeling, 25(4), 621–638. <doi:10.1080/10705511.2017.1402334> The workflow implemented in this package follows the recommendations outlined in Van Der Nest et al. (2020). "An Overview of Mixture Modeling for Latent Evolutions in Longitudinal Data: Modeling Approaches, Fit Statistics, and Software." Advances in Life Course Research, 43, Article 100323. <doi:10.1016/j.alcr.2019.100323>.

Depends R (>= 4.1.0),

License GPL (>= 3)

Imports MplusAutomation, magrittr, tibble, dplyr, tidyr, tidyselect, stringr, purrr, ggplot2, glue, parallel

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://github.com/OlivierPDS/MplusLGM

BugReports https://github.com/OlivierPDS/MplusLGM/issues

LazyData true

Config/pak/sysreqs texlive libicu-dev libssl-dev

Repository https://olivierpds.r-universe.dev

RemoteUrl https://github.com/olivierpds/mpluslgm

RemoteRef HEAD

RemoteSha b3bc8072d8c27eeeec9f93a5b60dba622a33c27f

Contents

`fitGBTM` *Fit Group-Based Trajectory Models (GBTM) for class enumeration.*

Description

Perform class enumeration by fitting a series of GBTM in Mplus across a predetermined range of classes, and returning a list of fitted models for evaluation and comparison.

Usage

```
fitGBTM(
  data,
  outvar,
  catvar = FALSE,
  idvar,
  min_k = 2L,
  max_k = 6L,
  starting_val = 500,
  polynomial = 1,
  timescores,
  timescores_indiv = FALSE,
  estimator = c("MLR", "ML", "WLSMV", "WLS"),
  transformation = c("LOGIT", "PROBIT"),
  output = c("TECH1", "TECH11", "SAMPSTAT", "STANDARDIZED"),
  plot = "PLOT3",
  save = "FSCORES",
  wd = "Results"
)
```

Arguments

data	A data frame containing all variables for the trajectory analysis.
outvar	A character vector specifying the outcome variables at different times.
catvar	A logical value indicating whether the outcome variable is categorical. Default is FALSE.
idvar	A character string specifying the ID variable.
min_k	An integer specifying the minimum number of latent classes to evaluate. Default is 2.
max_k	An integer specifying the maximum number of latent classes to evaluate. Default is 6.
starting_val	A numeric value specifying the number of random starting values to generate for the initial optimization stage. Note that the number of final stage optimizations will be set as equal to half of this value.
polynomial	An integer specifying the order of the polynomial used to model trajectories. Supported values are: 1 (linear), 2 (quadratic), 3 (cubic). Default is 1.
timescores	A numeric vector specifying the time scores for the model. If timescores_indiv = TRUE, a character vector should be used to specify variables with individually varying times of observation.
timescores_indiv	A logical value indicating whether to use individually varying times of observation for the outcome variable. Default is FALSE.
estimator	A character string to specify the estimator to use in the analysis. Default is 'MLR'.
transformation	A character string to specify the latent response variable transformation to use when the outcome variable is categorical. Default is LOGIT.
output	A character vector specifying the requested Mplus output options for the model.
plot	A character string specifying the requested Mplus plot options for the model. Default is PLOT3.
save	A character string specifying the type of results to be saved by Mplus. Default is FSCORES.
wd	A character string specifying the directory where the results folder will be created for saving Mplus input, output, and data files. Default is the current working directory.

Details

The fitGBTM function automates the process of fitting GBTM, iterating through an increasing number of class. This function is designed for conducting class enumeration and help identifying the optimal number of latent classes. GBTM should converge the quickest to a solution given its lower number of free parameters when compared to other LGM.

The function operates as follows:

- 1. Iterate over an increasing number of classes, ranging from min_k to max_k.

- 2. Create GBTM `mplusObject` with appropriate class specification using the `LGMobject` function.
- 3. Fit models using the `runLGM` function, ensuring convergence by increasing the number of random starting values until the best log-likelihood is replicated.
- 4. Return a list of `mplusObject` including results for the fitted GBTM models with each class structures.

The function automates the procedure outlined for model selection in: Van Der Nest et al,. (2020). "An overview of mixture modelling for latent evolutions in longitudinal data: Modelling approaches, fit statistics and software." Advances in Life Course Research 43: 100323.

This function builds upon the capabilities of the `mplusObject` and `mplusModeler` functions from the MplusAutomation package.

Value

A list of `mplusObject` including the fitted GBTM models for each class specification.

See Also

`LGMobject` for creating the mplusObject of a latent growth model. `runLGM` for conducting latent growth modelling with an mplusObject.

Examples

```
# Example usage:
GBTM_models <- fitGBTM(
  data = symptoms,
  outvar = paste("sx", seq(from = 0, to = 24, by = 6), sep = "_"),
  catvar = FALSE,
  idvar = "id",
  starting_val = 500,
  min_k = 2L,
  max_k = 6L,
  timescores = seq(from = 0, to = 24, by = 6),
  timescores_indiv = FALSE,
  polynomial = 1,
  output = c("TECH1", 'TECH14', "SAMPSTAT", "STANDARDIZED"),
  plot = "PLOT3",
  save = "FSCORES",
  wd = file.path("Results", "Trajectories")
)

# Accessing the model:
GBTM2 <- GBTM_models[[1]] #with 2 latent classes
GBTM3 <- GBTM_models[[2]] #with 3 latent classes
GBTM4 <- GBTM_models[[3]] #with 4 latent classes
```

fitGCM *Fit Growth Curve Models (GCM)*

Description

Customize and execute GCM in Mplus, offering flexibility in model configuration and parameter estimation.

Usage

```
fitGCM(
  data,
  outvar,
  catvar = FALSE,
  idvar,
  starting_val = 500,
  polynomial = 1,
  timescores = timescores,
  timescores_indiv = FALSE,
  estimator = c("MLR", "ML", "WLSMV", "WLS"),
  transformation = c("LOGIT", "PROBIT"),
  output = c("TECH1", "SAMPSTAT", "STANDARDIZED"),
  plot = "PLOT3",
  save = "FSCORES",
  wd = "Results"
)
```

Arguments

`data`	A data frame containing all variables for the trajectory analysis.
`outvar`	A character vector specifying the outcome variables at different times.
`catvar`	A logical value indicating whether the outcome variable is categorical. Default is `FALSE`.
`idvar`	A character string specifying the ID variable.
`starting_val`	A numeric value specifying the number of random starting values to generate for the initial optimization stage. Note that the number of final stage optimizations will be set as equal to half of this value.
`polynomial`	An integer specifying the order of the polynomial used to model trajectories. Supported values are: 1 (linear), 2 (quadratic), 3 (cubic). Default is 1.
`timescores`	A numeric vector specifying the time scores for the model. If `timescores_indiv = TRUE`, a character vector should be used to specify variables with individually varying times of observation.
`timescores_indiv`	A logical value indicating whether to use individually varying times of observation for the outcome variable. Default is `FALSE`.

estimator	A character string to specify the estimator to use in the analysis. Default is 'MLR'.
transformation	A character string to specify the latent response variable transformation to use when the outcome variable is categorical. Default is LOGIT.
output	A character vector specifying the requested Mplus output options for the model.
plot	A character string specifying the requested Mplus plot options for the model.
save	A character string specifying the type of results to be saved by Mplus.
wd	A character string specifying the directory where the results folder will be created for saving Mplus input, output, and data files. Default is the current working directory.

Details

The fitGCM function automates the process of specifying, customizing and fitting GCM in Mplus.

This function builds upon the capabilities of the mplusObject and mplusModeler functions from the MplusAutomation package.

Value

A list of class mplusObjects including results for the fitted GCM.

See Also

LGMobject for creating the mplusObject of a latent growth model. runLGM for conducting latent growth modelling with an mplusObject.

Examples

```
# Example usage:
GCM_model <- fitGCM(
  data = symptoms,
  outvar = paste("sx", seq(from = 0, to = 24, by = 6), sep = "_"),
  catvar = FALSE,
  idvar = "id",
  starting_val = 500,
  polynomial = 3,
  timescores = seq(from = 0, to = 24, by = 6),
  timescores_indiv = FALSE,
  output = c("TECH1", "SAMPSTAT", "STANDARDIZED"),
  plot = "PLOT3",
  save = "FSCORES",
  wd = file.path("Results", "Trajectories")
  )
```

`fitLCGA` *Fit Latent Class Growth Analysis (LCGA) models to refine covariance structure.*

Description

Refine the residual variance structure by fitting a series of LCGA models progressively allowing for the dependence of residuals on time and/or class, and returning a list of fitted models for evaluation and comparison.

Usage

```
fitLCGA(
  data,
  outvar,
  catvar = FALSE,
  idvar,
  k,
  starting_val = 500,
  polynomial = 1,
  timescores,
  timescores_indiv = FALSE,
  estimator = c("MLR", "ML", "WLSMV", "WLS"),
  transformation = c("LOGIT", "PROBIT"),
  output = c("TECH1", "TECH11", "SAMPSTAT", "STANDARDIZED"),
  plot = "PLOT3",
  save = "FSCORES",
  wd = "Results"
)
```

Arguments

`data`	A data frame containing all variables for the trajectory analysis.
`outvar`	A character vector specifying the outcome variables at different times.
`catvar`	A logical value indicating whether the outcome variable is categorical. Default is `FALSE`.
`idvar`	A character string specifying the ID variable.
`k`	An integer specifying the number of latent classes for the model.
`starting_val`	A numeric value specifying the number of random starting values to generate for the initial optimization stage. Note that the number of final stage optimizations will be set as equal to half of this value.
`polynomial`	An integer specifying the order of the polynomial used to model trajectories. Supported values are: 1 (linear), 2 (quadratic), 3 (cubic). Default is 1.
`timescores`	A numeric vector specifying the time scores for the model. If `timescores_indiv = TRUE`, a character vector should be used to specify variables with individually varying times of observation.

`timescores_indiv`
: A logical value indicating whether to use individually varying times of observation for the outcome variable. Default is `FALSE`.

`estimator`
: A character string to specify the estimator to use in the analysis. Default is 'MLR'.

`transformation`
: A character string to specify the latent response variable transformation to use when the outcome variable is categorical. Default is `LOGIT`.

`output`
: A character vector specifying the requested Mplus output options for the model.

`plot`
: A character string specifying the requested Mplus plot options for the model.

`save`
: A character string specifying the type of results to be saved by Mplus.

`wd`
: A character string specifying the directory where the results folder will be created for saving Mplus input, output, and data files. Default is the current working directory.

Details

The `fitLCGA` function automates the process of fitting LCGA models, iterating through 3 varying residual variance specifications:

- - Relaxed residual variance across time
- - Relaxed residual variance across class
- - Relaxed residual variance across both time and class

This function is designed to help identify the optimal residual variance structure while examining convergence issues as model complexity increases.

The function operates as follows:

- 1. Iterate over the 3 residual variance specifications
- 2. Create LCGA `mplusObject` with appropriate residual variance specification using the `LGMobject` function.
- 3. Fit models using the `runLGM` function, ensuring convergence by increasing the number of random starting values until the best log-likelihood is replicated.
- 4. Return a list of `mplusObject` including results for the fitted LCGA models with each residual variance structures

The function automates the procedure outlined for model selection in: Van Der Nest et al., (2020). "An overview of mixture modelling for latent evolutions in longitudinal data: Modelling approaches, fit statistics and software." Advances in Life Course Research 43: 100323.

This function builds upon the capabilities of the `mplusObject` and `mplusModeler` functions from the MplusAutomation package.

Value

A list of `mplusObject` including results for the fitted LCGA models.

See Also

LGMobject for creating the mplusObject of a latent growth model. runLGM for conducting latent growth modelling with an mplusObject.

Examples

```
# Example usage:
LCGA_models <- fitLCGA(
 data = symptoms,
 outvar = paste('sx', seq(from = 0, to = 24, by = 6), sep = "_"),
 catvar = FALSE,
 idvar = "id",
 starting_val = 500,
 k = 3L,
 timescores = seq(from = 0, to = 24, by = 6),
 timescores_indiv = FALSE,
 polynomial = 3,
 output = c('TECH1', 'TECH14', 'SAMPSTAT', 'STANDARDIZED'),
 wd = file.path('Results', 'Trajectories')
)

# Accessing the models:
LCGA_t <- LCGA_models[[1]] #with relaxed residual variance across time
LCGA_c <- LCGA_models[[2]] #with relaxed residual variance across class
LCGA_tc <- LCGA_models[[3]] #with relaxed residual variance across time and class
```

---

getBest *Select best-fitting model from a list of Latent Growth Models (LGM)*

Description

Identify and extract the best-fitting model from a list of LGM based on a specified set of criteria applied to a summary table of the models fit indices.

Usage

```
getBest(
  lgm_object,
  ic = c("BIC", "aBIC", "AIC", "CAIC", "AICC"),
  lrt = c("none", "aLRT", "BLRT"),
  p = 0.05
)
```

Arguments

lgm_object	A list of LGM mplusObject to evaluate.

ic	A character string specifying the information criterion (IC) to use for selecting the best-fitting model. Supported options are Bayesian Information Criterion (BIC), sample-size-adjusted BIC (aBIC), Akaike Information Criterion (AIC), Consistent Akaike Information Criterion (CAIC), and AIC corrected (AICC). The default is BIC.
lrt	A character string specifying the likelihood ratio test (LRT) to use for selecting the best-fitting model. Supported options are Bootstrap LRT (BLRT) and Lo-Mendel-Rubin adjusted LRT (aLRT). Default is "none", the selection of the the best-fitting model is only made based on the selected IC.
p	A numeric value specifying the p-value threshold for statistical significance when using LRT-based selection of the best-fitting model. Default is 0.05.

Details

The function select the best-fitting model based on the following criteria:

- 1. Models with convergence errors are excluded.
- 2. The model with the lowest information criterion (IC) is selected.
- 3. If specified, the likelihood ratio test (LRT) is used to determine whether the K-class model can be reduced to K-1 classes.
- 4. The resulting model throw a warning if it meet any of the following conditions:
  - - Entropy is below 0.5.
  - - Any class has an average posterior probability of assignment (APPA) below 0.7.
  - - Any class represents less than 5% of the sample size.

Value

The LGM `mplusObject` of the best-fitting model.

Examples

```
# Example usage:
GBTM_best <- getBest(
  lgm_object = GBTM_models,
  ic = "BIC",
  lrt = "aLRT",
  p = 0.05
)

best_fit <- getFit(GBTM_best)

print(best_fit)
```

getFit *Get fit indices from Latent Growth Models (LGM)*

Description

Extract key information from Mplus LGM objects, including model summaries, fit statistics, class details, warnings, and errors. The function accounts for non-converging models and compiles the extracted information into a single data frame to facilitate model evaluation and comparison.

Usage

```
getFit(lgm_object)
```

Arguments

lgm_object	A single LGM mplusObject or a list of LGM mplusObject (nested lists supported).

Details

- - Model summaries such as the title, log-likelihood value and number of observations, parameters and latent classes.
- - Model fit indices such as the BIC, aBIC, AIC, AICC and CAIC along with statistics from the BLRT and adjusted LMR-LRT, if requested.
- - Latent class counts and proportions.
- - Classification confidence measures such as the average posterior probabilities (APPA) and entropy.
- - Mplus warnings or errors encountered during model estimation.

This output facilitates side-by-side comparison of models to support model evaluation and selection.

Value

A data frame with a row for each LGM of the input list.

Examples

```
# Example usage:
fit_indices <- getFit(lgm_object = GCM_model)
fit_indices <- getFit(lgm_object = list(GCM_model, GBTM_models, LCGA_models))

print(fit_indices)
```

getPoly *Refine Polynomial Order in Latent Growth Modelling (LGM)*

Description

Refine the polynomial order for each class of a LGM by iteratively removing non-significant growth factors, and running the updated models.

Usage

```
getPoly(lgm_object, wd = "Results")
```

Arguments

lgm_object	A LGM mplusObject, typically generated with the LGMobject and runLGM functions.
wd	A character string specifying the directory where the results folder will be created for saving Mplus input, output, and data files. Default is the current working directory.

Details

The getPoly function refines the polynomial order of a LGM mplusObject through an iterative process. In addition to ensuring the statistical significance of growth factors in each latent class, the function ensure that the best loglikelihood value of the updated model is replicated.

The function works as follows:

- 1. Extract model information from the provided LGM mplusObject.
- 2. Evaluate the statistical significance of the highest-order growth factor in each class.
- 3. Remove non-significant growth factors (p-value > 0.05) from the model.
- 4. Update the LGM mplusObject to reflect changes in the growth factor structure.
- 5. Re-run the updated mplusObject until log-likelihood values are successfully replicated using the runLGM function.
- 6. Repeat the process until the highest-order growth factor of all classes are statistically significant or reduce to intercept-only.

The function automates the procedure outlined for model selection in: Van Der Nest et al,. (2020). "An overview of mixture modelling for latent evolutions in longitudinal data: Modelling approaches, fit statistics and software." Advances in Life Course Research 43: 100323.

Value

A LGM mplusObject including the results of the updated model with the refined polynomial order.

See Also

LGMobject for creating the mplusObject of a latent growth model. runLGM for conducting latent growth modelling with a mplusObject.

Examples

```
# Example usage:
final_model <- getPoly(
  lgm_object = LCGA_best,
  wd = "Results"
  )

final_fit <- getFit(final_model)

 print(final_fit)
```

---

getSpaghetti *Plot individual trajectories of outcome - Spaghetti plot*

Description

Generate a spaghetti plot to visualize the individual trajectories of a given outcome across time..

Usage

```
getSpaghetti(data, outvar)
```

Arguments

data	A data frame containing all variables for the trajectory analysis.
outvar	A character vector specifying the outcome variables at different times.

Value

A ggplot object displaying the spaghetti plot of individual trajectories.

Examples

```
# Example usage:
plot <- getSpaghetti(
  data = symptoms,
  outvar = paste("sx", seq(from = 0, to = 24, by = 6), sep = "_"))

print(plot)
```

LGMobject *Create Mplus model objects for Latent Growth Modelling (LGM)*

Description

Provide flexibility for specifying Mplus LGM objects with various latent class and residual variance structures, and capturing individual differences in growth trajectories. Support Growth Curve Models (GCM), Growth-Based Trajectory Models (GBTM) and Latent Class Growth Analysis (LCGA). Once created, the model can be estimated using the runLGM function.

Usage

```
LGMobject(
  data,
  outvar,
  catvar = FALSE,
  idvar,
  k,
  starting_val,
  estimator = c("MLR", "ML", "WLSMV", "WLS"),
  transformation = c("LOGIT", "PROBIT"),
  lgm_type = c("gcm", "gbtm", "lcga_t", "lcga_c", "lcga_tc"),
  polynomial = 1,
  timescores,
  timescores_indiv = FALSE,
  output,
  plot,
  save
)
```

Arguments

data	A data frame containing all variables for the trajectory analysis.
outvar	A character vector specifying the outcome variables at different times.
catvar	A logical value indicating whether the outcome variable is categorical. Default is FALSE.
idvar	A character string specifying the ID variable.
k	An integer specifying the number of latent classes for the model.
starting_val	A numeric value specifying the number of random starting values to generate for the initial optimization stage. Note that the number of final stage optimizations will be set as equal to half of this value.
estimator	A character string to specify the estimator to use in the analysis. Default is 'MLR'.
transformation	A character string to specify the latent response variable transformation to use when the outcome variable is categorical. Default is LOGIT.

`lgm_type` A character string specifying the residual variance structure of the growth model. Options include:

- - "gcm" (relaxed residual variance across time),
- - "gbtm" (fixed residual variance across time and class),
- - "lcga_t" (relaxed residual variance across time),
- - "lcga_c" (relaxed residual variance across class),
- - "lcga_tc" (relaxed residual variance across both time and class).

`polynomial` An integer specifying the order of the polynomial used to model trajectories. Supported values are: 1 (linear), 2 (quadratic), 3 (cubic). Default is 1.

`timescores` A numeric vector specifying the time scores for the model. If `timescores_indiv = TRUE`, a character vector should be used to specify variables with individually varying times of observation.

`timescores_indiv` A logical value indicating whether to use individually varying times of observation for the outcome variable. Default is `FALSE`.

`output` A character vector specifying the requested Mplus output options for the model.

`plot` A character string specifying the requested Mplus plot options for the model.

`save` A character string specifying the type of results to be saved by Mplus.

Details

The `LGMobject` function facilitates and automates the appropriate model specification for conducting latent growth modeling in Mplus. It creates the relevant sections of an Mplus input file, including: TITLE, VARIABLE, ANALYSIS, MODEL, OUTPUT, PLOT, and SAVEDATA.

This function builds upon the capabilities of the `mplusObject`function from the MplusAutomation package.

Value

A list of class `mplusObject` with elements specifying sections of an Mplus input file for conducting latent growth modeling.

See Also

`mplusObject` for creating an mplusObject. `runLGM` for conducting latent growth modelling with an mplusObject.

Examples

```
# Example usage:
GBTM_object <- LGMobject(
  data = symptoms,
  outvar = paste("sx", seq(from = 0, to = 24, by = 6), sep = "_"),
  idvar = "id",
  catvar = FALSE,
  k = 3L,
  starting_val = 500,
```

```
lgm_type = "gbtm",
polynomial = 3,
timescores = seq(from = 0, to = 24, by = 6),
timescores_indiv = FALSE,
output = c("TECH1", "TECH14", "SAMPSTAT", "STANDARDIZED"),
plot = "PLOT3",
save = "FSCORES"
)
```

`runLGM` *Run Latent Growth Models (LGM) and replicate the best loglikelihood value (LL)*

Description

Run iterations of an LGM, doubling the number of starting values until the best LL value has replicated at least twice, both within and between models.

Usage

```
runLGM(lgm_object, wd)
```

Arguments

`lgm_object`	An `mplusObject` with predefined random starting values (`STARTS`) in the ANALYSIS section.
`wd`	A character string specifying the directory where the results folder will be created for saving the Mplus input, output, and data files. Default is the current working directory.

Details

The `runLGM` function run iterations of an LGM in Mplus while gradually increasing the number of random starting values used to optimize the loglikelihood. This approach aims to prevent estimation issues related to local maxima, which can result in selecting the inappropriate model during class enumeration. The function works as follows:

- 1. Estimate the model using the predefined number of random starting values.
- 2. Rerun the model with double the number of starting values.
- 3. Continue until the best LL value is successfully replicated both within the model and between 2 consecutive model run, or the maximum number of allowed starting values is reached. By default the maximum number of allowed starting values is set 2 times the number of initial starting values raised to the power of 5.
- 4. Return the `mplusObject` from the replicated model.

This function builds upon the capabilities of the `mplusModeler` function from the MplusAutomation package.

Value

A list of class `mplusObject` including results for the replicated model, alongside with :

- - The Mplus input and data files used for the model.
- - The output files generated by Mplus.
- - The data results files saved by Mplus.

See Also

`mplusModeler` for running, and reading an mplusObject. `LGMobject` for creating the mplusObject for a latent growth model.

Examples

```
# Example usage:
GBTM_model <-  runLGM(
  lgm_object = GBTM_object,
  wd = file.path("Results", "Trajectories"))
```

---

symptoms *Symptoms Data*

Description

A simulated, longitudinal dataset capturing symptom severity with an arbitrary scale (total score: 0-28), over a 2-year follow-up period for 350 individuals. The data is not normally distributed and exhibits heterogeneity, including latent (unobserved) trajectories of symptom severity that reflect diverse progression patterns across individuals. The dataset contains no missing data.

Usage

```
symptoms
```

Format

A dataframe with 1 row per individual, 350 observations and 10 variables.

id Individual identifier, numeric.

sx_0 Symptoms severity at month 0, numeric.

sx_3 Symptoms severity at month 3, numeric.

sx_6 Symptoms severity at month 6, numeric.

sx_9 Symptoms severity at month 9, numeric.

sx_12 Symptoms severity at month 12, numeric.

sx_15 Symptoms severity at month 15, numeric.

sx_18 Symptoms severity at month 18, numeric.

sx_21 Symptoms severity at month 21, numeric.

sx_24 Symptoms severity at month 24, numeric.

Source

Data simulated using modgo

Index