Package: slca (via r-universe)

October 30, 2026

Type Package

Title Structural Modeling for Multiple Latent Class Variables

Version 1.4.3

Author Youngsun Kim [aut, cre] (ORCID: <https://orcid.org/0000-0001-8003-1939>), Hwan Chung [aut] (ORCID: <https://orcid.org/0000-0002-8969-9086>)

Maintainer Youngsun Kim <yskstat@gmail.com>

Description Provides comprehensive tools for the implementation of Structural Latent Class Models (SLCM), including Latent Transition Analysis (LTA; Linda M. Collins and Stephanie T. Lanza, 2009) <doi:10.1002/9780470567333>, Latent Class Profile Analysis (LCPA; Hwan Chung et al., 2010) <doi:10.1111/j.1467-985x.2010.00674.x>, and Joint Latent Class Analysis (JLCA; Saebom Jeon et al., 2017) <doi:10.1080/10705511.2017.1340844>, and any other extended models involving multiple latent class variables. Details on the methodology and implementation are given in Kim and Chung (2026) <doi:10.18637/jss.v117.i08>.

License GPL (>= 3)

URL https://kim0sun.github.io/slca/

BugReports https://github.com/kim0sun/slca/issues

Depends R (>= 2.10)

Imports DiagrammeR, magrittr, MASS, Rcpp, stats

Suggests testthat (>= 3.1.7)

LinkingTo Rcpp

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Encoding UTF-8

LazyData true

NeedsCompilation yes

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

Date/Publication 2026-10-30 19:34:59 UTC

RemoteUrl https://github.com/cran/slca

RemoteRef HEAD

RemoteSha 8ef019ec698230d0daf28e5d514fe6ca5dd0de20

Contents

addhealth . . . 2
compare . . . 4
confint.slcafit . . . 6
estimate . . . 7
gof . . . 9
gss7677 . . . 11
nlsy_jlcpa . . . 12
nlsy97 . . . 13
param . . . 15
predict.slcafit . . . 16
regress . . . 17
reorder.slcafit . . . 19
simulate.slca . . . 20
slca . . . 22
slcaControl . . . 24

Index **26**

`addhealth` *Adolescent Depression Data from the Add Health Study*

Description

This dataset contains responses from the National Longitudinal Study of Adolescent Health (Add Health), focusing on adolescents' experiences with depression. The subjects, who were in Grades 10 and 11 during the 1994–1995 academic year, provided data on at least one measure of adolescent delinquency in Wave I.

These data can be used to replicate the latent class analysis conducted by Collins and Lanza (2009). The dataset includes five covariates, notably grade level and sex of respondents, along with variables capturing depressive emotions: sadness (`S1-S4`), feeling disliked (`D1-D2`), and feelings of failure (`F1-F2`).

Responses for these variables were initially categorized as "Never," "Sometimes," "Often," or "Most or All of the Time." In this dataset, responses have been recoded as "No" for "Never" and "Yes" for all other responses, providing a longitudinal perspective on adolescent depression across Waves I and II. Variables with the suffix `"w1"` are from Wave I, while those with the suffix `"w2"` are from Wave II.

Usage

```
addhealth
```

Format

A data frame with 2061 rows and 18 variables:

`GRADE` Respondent's grade level at Wave I.

`SEX` Respondent's sex
levels: (1)`Male`, (2)`Female`.

`S1w1`, `S1w2` I felt that I could not shake off the blues even with help from my family and friends.

`S2w1`, `S2w2` I felt depressed.

`S3w1`, `S3w2` I felt lonely.

`S4w1`, `S4w2` I felt sad.

`D1w1`, `D1w2` People were unfriendly to me.

`D2w1`, `D2w2` I felt that people disliked me

`F1w1`, `F1w2` I thought my life had been a failure.

`F2w1`, `F2w2` I felt life was not worth living

Source

https://addhealth.cpc.unc.edu/data/#public-use

References

Collins, L.M., & Lanza, S.T. (2009). Latent Class and Latent Transition Analysis: With Applications in the Social, Behavioral, and Health Sciences.

J.R. Udry. The National Longitudinal Study of Adolescent Health (Add Health), Waves I & II, 1994-1996. Carolina Population Center, University of North Carolina at Chapel Hill, Chapel Hill, NC, 2003.

Examples

```
library(magrittr)
data <- addhealth[1:200,]
lta5 <- slca(
   DEP1(5) ~ S1w1 + S2w1 + S3w1 + S4w1 + D1w1 + D2w1 + F1w1 + F2w1,
   DEP2(5) ~ S1w2 + S2w2 + S3w2 + S4w2 + D1w2 + D2w2 + F1w2 + F2w2,
   DEP1 ~ DEP2
) %>% estimate(data, control = list(em.tol = 1e-6))
lta5inv <- slca(
   DEP1(5) ~ S1w1 + S2w1 + S3w1 + S4w1 + D1w1 + D2w1 + F1w1 + F2w1,
   DEP2(5) ~ S1w2 + S2w2 + S3w2 + S4w2 + D1w2 + D2w2 + F1w2 + F2w2,
   DEP1 ~ DEP2,
   constraints = c("DEP1", "DEP2")
) %>% estimate(data, control = list(em.tol = 1e-6))

compare(lta5inv, lta5, test = "chisq")
lta5inv %>% param()
```

compare *Compare Two Fitted* slca *Models*

Description

Conducts a relative model fit test between two fitted SLCM models using the deviance statistic.

Usage

```
compare(
  model1,
  model2,
  test = c("none", "chisq", "boot"),
  nboot = 50,
  method = c("hybrid", "em", "nlm"),
  plot = FALSE,
  maxiter = 1000,
  tol = 1e-08,
  verbose = FALSE,
  nrep = 3
)
```

Arguments

model1	an object of class slcafit.
model2	another object of class slcafit to be compared with model1.
test	a character string specifying the type of test to be conducted. If "chisq", a chi-squared test is conducted. If "boot", a bootstrap test is conducted.
nboot	a positive number specifying the number of bootstrap iterations to perform (used only when test = "boot"). Non-integer values are rounded up. The default is 50.
method	a character string specifying the estimation method for bootstrapping.
plot	a logical value indicating whether to display a histogram of G-squared statistics for the bootstrap samples (applicable only for test = "boot"). The default is FALSE.
maxiter	an integer specifying the maximum number of iterations allowed during each bootstrap estimation round. The default is 1000.
tol	numeric value setting the convergence tolerance for each bootstrap iteration. The default is 1e-8.
verbose	a logical value indicating whether to print progress updates on completed bootstrap iterations. The default is FALSE.
nrep	a positive number specifying the number of starting values used for each model in a bootstrap iteration. The fitted-model parameters are used first and additional starting values are generated randomly. The default is 3.

Value

A `data.frame` containing the number of parameters (Df), loglikelihood, AIC, BIC, G-squared statistics, and the residual degree of freedom for each object. If a statistical test is conducted (via `test`), the resulting p-value for the comparison is also included. For `test = "boot"`, failed bootstrap replicates are omitted from the p-value calculation and recorded in the `bootFail`, `boot.fail.msgs`, and `boot.all.msgs` attributes.

Bootstrap samples retain the fitted data's missing-value pattern. Nonconverged refits and negative bootstrap likelihood ratios are counted as failures. Hybrid refits must converge in their final `nlm` stage. With B successful replicates and b statistics at least as large as observed, the p-value is $(b + 1)/(B + 1)$. When the observed likelihood ratio is effectively zero, the p-value is one; no bootstrap samples or histogram are generated.

See Also

gof

Examples

```
library(magrittr)
set.seed(20261017)
data <- gss7677[gss7677$COHORT == "YOUNG", ]
stat2 <- slca(status(2) ~ PAPRES + PADEG + MADEG) %>%
   estimate(data = data, control = list(verbose = FALSE))
stat3 <- slca(status(3) ~ PAPRES + PADEG + MADEG) %>%
   estimate(data = data, control = list(verbose = FALSE))
stat4 <- slca(status(4) ~ PAPRES + PADEG + MADEG) %>%
   estimate(data = data, control = list(verbose = FALSE))

gof(stat2, stat3, stat4)
gof(stat2, stat3, stat4, test = "chisq")

# Five replicates illustrate the interface only; use more for inference.
absolute_boot <- gof(stat2, stat3, stat4, test = "boot",
                     nboot = 5, maxiter = 5000, tol = 1e-6)
absolute_boot
attr(absolute_boot, "bootFail")


compare(stat3, stat4)
compare(stat3, stat4, test = "chisq")

# Use EM refits here; always inspect convergence failures.
relative_boot <- compare(stat3, stat4, test = "boot", method = "em",
                         nboot = 5, maxiter = 5000, tol = 1e-6)
relative_boot
attr(relative_boot, "bootFail")
```

`confint.slcafit` *Confidence Intervals for Model Parameters*

Description

Computes confidence intervals for one or more parameters of a fitted model.

Usage

```
## S3 method for class 'slcafit'
confint(object, parm, level = 0.95, type = c("probs", "logit"), ...)
```

Arguments

`object`	an object of class `slcafit`.
`parm`	an integer or string specifying the parameters for which confidence intervals are to be computed.
`level`	a numeric value representing the confidence level for the intervals. The default is `0.95` (95% confidence level).
`type`	a character string specifying the format in which the results should be returned. Options include `"probs"` for probability format and `"logit"` for log-odds (logit) format, with the default being `"probs"`.
`...`	additional arguments.

Value

A `matrix` with two columns representing the confidence intervals for the selected parameters. The column names correspond to the specified confidence level:

- `100 * (level / 2)%`: The lower bound of the confidence interval.
- `100 * (1 - level / 2)%`: The upper bound of the confidence interval.

The `level` argument determines the confidence level, with common values being `0.95` for a 95% confidence interval and `0.99` for a 99% confidence interval.

Examples

```
param(nlsy_jlcpa, index = TRUE)
confint(nlsy_jlcpa)
confint(nlsy_jlcpa, 1:4)
```

`estimate` *Estimate Parameters of an* `slca` *Object*

Description

Estimates the parameters of a model created using the `slca` function.

Usage

```
estimate(x, ...)

## S3 method for class 'slca'
estimate(x,
   data,
   method = c("em", "hybrid", "nlm"),
   fix2zero = NULL,
   control = slcaControl(), ...)
```

Arguments

`x`	an `slca` object defining the `slca` model to be estimated.
`...`	additional arguments passed to the estimation process.
`data`	a `data.frame` containing the observed categorical variables included in the model.
`method`	a character string specifying the estimation method for SLCM parameters. The default is `"em"`, which uses the expectation-maximization (EM) algorithm. The alternative `"nlm"` employs the Newton-Raphson algorithm via the `nlm` function, while `"hybrid"` combines both approaches, starting with EM and finishing with `nlm` for refined estimates.
`fix2zero`	a `vector` specifying parameters to be constrained to zero. When re-estimating a fitted object, an omitted argument retains its existing constraints; explicitly supplying `NULL` removes the constraint specification. See the 'Details' section for further information.
`control`	a `list` of control parameters for the estimation procedure. Modify default values using `slcaControl()`.

Details

The `fix2zero` argument allows you to constrain specific parameters to zero. Each parameter is associated with a unique index, which can be identified using the param function with the argument `index = TRUE`. To apply constraints, provide the relevant parameter indices in the `fix2zero` arguments with vector.

Manifest responses are categorical; continuous-response models are not supported. Corresponding indicators under equality constraints must have identical factor levels, including labels and order. For categories absent from a sample, declare the full levels explicitly, for example `factor(x, levels = 1:3)`, consistently across constrained indicators.

Value

An object of class `slcafit` containing the following components:

`model`	a `list` describing of the model structure.
`method`	the estimation method used.
`arg`	a brief description of the model used during estimation.
`mf`	the `data.frame` used for estimation.
`par`	the log of the estimated paramters.
`logit`	the log-odds of the estimated parameters.
`score`	the score function for the estimated parameters.
`posterior`	a `list` of posterior probablities for each latent class variable.
`convergence`	a named logical vector indicating whether convergence was achieved for the EM and `nlm` steps. Steps not used by the selected method are returned as `NA`.
`loglikelihood`	the loglikelihood value of the estimated model.
`control`	the control settings used during the estimation process.

The returned object can be further processed using the param function to extract the estimated parameters or their standard errors. The regress function allows for logistic regression analysis using a three-step approach to evaluate the effects of external variables on latent class variables. Additionally, several other methods are available, including predict.slcafit, reorder.slcafit, gof, and others.

See Also

`slca()` `param()` `slcaControl()`

Examples

```
m <- slca(lc[3] ~ y1 + y2 + y3 + y4)
pi <- rep(1 / 3, 3)
rho <- c(.9, .1, .9, .1, .9, .1, .9, .1, # class 1
         .9, .1, .9, .1, .1, .9, .1, .9, # class 2
         .1, .9, .1, .9, .1, .9, .1, .9) # class 3
dt <- simulate(m, 200, parm = c(pi, rho))
estimate(m, dt$response)

# Several estimation methods
estimate(m, dt$response, method = "em",
         control = slcaControl(verbose = TRUE)) # default
estimate(m, dt$response, method = "nlm",
         control = slcaControl(verbose = TRUE))
estimate(m, dt$response, method = "hybrid",
         control = slcaControl(verbose = TRUE))

# Parameter restriction
mf <- estimate(m, dt$response)
param(mf, index = TRUE)
mf0 <- estimate(mf, fix2zero = c(4, 6, 8, 10))
```

```
param(mf0)

# Estimation control
estimate(m, dt$response, control = slcaControl(nrep = 3, verbose = TRUE))
estimate(m, dt$response, control = slcaControl(init.param = c(pi, rho)))
```

---

gof *Goodness-of-Fit Test for Fitted* slca *Model*

Description

Computes the AIC, BIC, and deviance statistic (G-squared) for assessing the goodness-of-fit of a fitted slca model. If the test argument is specified, absolute model fit can be evaluated using deviance statistics.

Usage

```
gof(object, ...)

## S3 method for class 'slcafit'
gof(
   object, ..., test = c("none", "chisq", "boot"),
   nboot = 100, plot = FALSE,
   maxiter = 5000, tol = 1e-6, verbose = FALSE
)

## S3 method for class 'slcafit'
gof(
  object,
  ...,
  test = c("none", "chisq", "boot"),
  nboot = 100,
  plot = FALSE,
  maxiter = 5000,
  tol = 1e-06,
  verbose = FALSE
)
```

Arguments

object	an object of class slcafit, or a list of slcafit objects. List names are used as model labels; unnamed elements are labeled [[1]], [[2]], and so on.
...	additional objects of class slcafit for comparison.
test	a character string specifying the type of test to be conducted. If "chisq", a chi-squared test is conducted. If "boot", a bootstrap test is conducted.
nboot	a positive number specifying the number of bootstrap rounds to be performed. Non-integer values are rounded up.

plot	a logical value indicating whether to print histogram of G-squared statistics for boostrap samples, only for test = "boot". The default is FALSE.
maxiter	an integer specifying the maximum number of iterations allowed for the estimation process during each bootstrap iteration. The default is 5000.
tol	a numeric value specifying the convergence tolerance for each bootstrap iteration. The default is 1e-6.
verbose	a logical value indicating whether to print progress updates on the number of bootstrapping rounds completed.

Value

A data.frame containing the number of parameters (Df), loglikelihood, AIC, BIC, G-squared statistics, and the residual degree of freedom for each object. If a statistical test is performed (using test), the result includes the corresponding p-value. For test = "boot", failed bootstrap replicates are omitted from the p-value calculation and recorded in the bootFail, boot.fail.msgs, and boot.all.msgs attributes. Absolute bootstrap samples retain the fitted data's missing-value pattern. Nonconverged refits are counted as failures. With B successful replicates and b statistics at least as large as observed, the bootstrap p-value is $(b + 1)/(B + 1)$.

See Also

compare

Examples

```
library(magrittr)
set.seed(20261017)
data <- gss7677[gss7677$COHORT == "YOUNG", ]
stat2 <- slca(status(2) ~ PAPRES + PADEG + MADEG) %>%
   estimate(data = data, control = list(verbose = FALSE))
stat3 <- slca(status(3) ~ PAPRES + PADEG + MADEG) %>%
   estimate(data = data, control = list(verbose = FALSE))
stat4 <- slca(status(4) ~ PAPRES + PADEG + MADEG) %>%
   estimate(data = data, control = list(verbose = FALSE))

gof(stat2, stat3, stat4)
gof(stat2, stat3, stat4, test = "chisq")

# Five replicates illustrate the interface only; use more for inference.
absolute_boot <- gof(stat2, stat3, stat4, test = "boot",
                     nboot = 5, maxiter = 5000, tol = 1e-6)
absolute_boot
attr(absolute_boot, "bootFail")


compare(stat3, stat4)
compare(stat3, stat4, test = "chisq")

# Use EM refits here; always inspect convergence failures.
relative_boot <- compare(stat3, stat4, test = "boot", method = "em",
```

```
                          nboot = 5, maxiter = 5000, tol = 1e-6)
relative_boot
attr(relative_boot, "bootFail")
```

---

gss7677 *GSS 1976-1977 Data on Social Status and Tolerance towards Minorities*

Description

This dataset contains responses from the General Social Survey (GSS) for the years 1976 and 1977, focusing on social status and tolerance towards minorities. The dataset can be used to replicate the analyses conducted in McCutcheon (1985) and Bakk et al. (2014).
It includes covariates such as interview year, age, sex, race, education level, and income. Social status-related variables include father's occupation and education level, as well as mother's education level. Tolerance towards minorities is measured by agreement with three questions: (1) allowing public speaking, (2) allowing teaching, and (3) allowing literature publication.

Usage

```
gss7677
```

Format

A data frame with 2942 rows and 14 variables:

`YEAR` Interview year (1976, 1977).

`COHORT` Respondent's age cohort.
Levels: (1) `YOUNG`, (2) `YOUNG-MIDDLE`, (4) `MIDDLE`, (5) `OLD`.

`SEX` Respondent's sex.
Levels: (1) `MALE`, (2) `FEMALE`.

`RACE` Respondent's race.
Levels: (1) `WHITE`, (2) `BLACK`, (3) `OTHER`.

`DEGREE` Respondent's education level.
Levels: (1) `LT HS`, (2) `HIGH-SCH`, (3) `HIGHER`.

`REALRINC` Respondent's income.

`PAPRES` Father's occupational prestige.
Levels: (1) `LOW`, (2) `MEDIUM`, (3) `HIGH`.

`PADEG` Father's education level.
Levels: (1) `LT HS`, (2) `HIGH-SCH`, (3) `COLLEGE`, (4) `BACHELOR`, (5) `GRADUATE`.

`MADEG` Mother's education level.
Levels: (1) `LT HS`, (2) `HIGH-SCH`, (3) `COLLEGE`, (4) `BACHELOR`, (5) `GRADUATE`.

`TOLRAC` Tolerance towards racists.

`TOLCOM` Tolerance towards communists.

TOLHOMO Tolerance towards homosexuals.

TOLATH Tolerance towards atheists.

TOLMIL Tolerance towards militarists.

Source

General Social Survey (GSS) 1976, 1977

References

Bakk Z, Kuha J. (2021) Relating latent class membership to external variables: An overview. Br J Math Stat Psychol. 74(2):340-362.

McCutcheon, A. L. (1985). A latent class analysis of tolerance for nonconformity in the American public. Public Opinion Quarterly, 49, 474–488.

Examples

```
library(magrittr)
gss500 <- gss7677[1:500,] %>% na.omit
model_stat <- slca(status(3) ~ PAPRES + PADEG + MADEG) %>%
   estimate(data = gss500, control = list(em.tol = 1e-6))
summary(model_stat)
param(model_stat)

model_tol <- slca(tol(4) ~ TOLRAC + TOLCOM + TOLHOMO + TOLATH + TOLMIL) %>%
   estimate(data = gss500, control = list(em.tol = 1e-6))
summary(model_tol)
param(model_tol)

model_lta <- slca(
   status(3) ~ PAPRES + PADEG + MADEG,
   tol(4) ~ TOLRAC + TOLCOM + TOLHOMO + TOLATH + TOLMIL,
   status ~ tol
) %>% estimate(data = gss500, control = list(em.tol = 1e-6))
summary(model_lta)
param(model_lta)


regress(model_lta, status ~ SEX, gss500)
regress(model_lta, status ~ SEX, gss500, method = "BCH")
regress(model_lta, status ~ SEX, gss500, method = "ML")
```

---

nlsy_jlcpa *JLCPA Model Estimated with NLSY97 Data*

Description

An slca model estimated using the NLSY97 dataset.

Usage

```
nlsy_jlcpa
```

Format

An `slcafit` object estimated for JLCPA model using `nlsy97` dataset.

References

Bureau of Labor Statistics, U.S. Department of Labor. National Longitudinal Survey of Youth 1997 cohort, 1997-2017 (rounds 1-18). Produced and distributed by the Center for Human Resource Research (CHRR), The Ohio State University. Columbus, OH: 2019.

Jeon, S., Seo, T. S., Anthony, J. C., & Chung, H. (2022). Latent Class Analysis for Repeatedly Measured Multiple Latent Class Variables. Multivariate Behavioral Research, 57(2–3), 341–355.

See Also

reorder.slcafit

`nlsy97` *NLSY97 Substance Use Data*

Description

This dataset contains substance use behavior data from the National Longitudinal Survey of Youth 1997 (NLSY97) for three years: 1998, 2003, and 2008. The dataset focuses on youth born in 1984 and tracks three types of substance use behaviors: tobacco/cigarette smoking, alcohol drinking, and marijuana use.

Usage

```
nlsy97
```

Format

A data frame with 1004 rows and 38 columns:

`SEX` Respondent's sex

`RACE` Respondent's race

`ESMK_98`, `ESMK_03`, `ESMK_08` (Ever smoked) Ever smoked in 1998, 2003, and 2008 (0: No, 1: Yes)

`FSMK_98`, `FSMK_03`, `FSMK_08` (Frequent smoke) Monthly smoking in 1998, 2003, and 2008 (0: No, 1: Yes)

`DSMK_98`, `DSMK_03`, `DSMK_08` (Daily smoke) Daily smoking in 1998, 2003, and 2008 (0: No, 1: Yes)

`HSMK_98`, `HSMK_03`, `HSMK_08` (Heavy smoke) 10+ cigarettes per day in 1998, 2003, and 2008 (0: No, 1: Yes)

EDRK_98, EDRK_03, EDRK_08 (Ever drunk) Ever drunk in 1998, 2003, and 2008? (0: No, 1: Yes)

CDRK_98, CDRK_03, CDRK_08 (Current drinker) Monthly drinking in 1998, 2003, and 2008 (0: No, 1: Yes)

WDRK_98, WDRK_03, WDRK_08 (Weakly drinker) 5+ days drinking in a month in 1998, 2003, and 2008 (0: No, 1: Yes)

BDRK_98, BDRK_03, BDRK_08 (Binge drinker) 5+ drinks on the same day at least one time in the last 30 day (0: No, 1: Yes)

EMRJ_98, EMRJ_03, EMRJ_08 (Ever marijuana used) Have you ever used marijuana in 1998, 2003, and 2008? (0: No, 1: Yes)

CMRJ_98, CMRJ_03, CMRJ_08 (Current marijuana user) Monthly marijuana use in 1998, 2003, and 2008 (0: No, 1: Yes)

OMRJ_98, OMRJ_03, OMRJ_08 (Occasional marijuana user) 10+ days marijuana use in a month in 1998, 2003, and 2008 (0: No, 1: Yes)

SMRJ_98, SMRJ_03, SMRJ_08 (School/work marijuana user) Marijuana use before/during school or work in 1998, 2003, and 2008 (0: No, 1: Yes)

Source

National Longitudinal Survey of Youth 1997 (NLSY97)

References

Bureau of Labor Statistics, U.S. Department of Labor. National Longitudinal Survey of Youth 1997 cohort, 1997-2017 (rounds 1-18). Produced and distributed by the Center for Human Resource Research (CHRR), The Ohio State University. Columbus, OH: 2019.

Examples

```
library(magrittr)
nlsy_smoke <- slca(SMK_98(3) ~ ESMK_98 + FSMK_98 + DSMK_98 + HSMK_98) %>%
   estimate(data = nlsy97, control = list(verbose = FALSE))
summary(nlsy_smoke)


# A reproducible subset and two-class models keep these syntax examples small.
# Choose class counts and fit the full data separately for substantive inference.
set.seed(20261017)
example_data <- nlsy97[sample.int(nrow(nlsy97), 500), ]
# JLCA
model_jlca <- slca(
   SMK_98(2) ~ ESMK_98 + FSMK_98 + DSMK_98 + HSMK_98,
   DRK_98(2) ~ EDRK_98 + CDRK_98 + WDRK_98 + BDRK_98,
   MRJ_98(2) ~ EMRJ_98 + CMRJ_98 + OMRJ_98 + SMRJ_98,
   SUB_98(2) ~ SMK_98 + DRK_98 + MRJ_98
) %>% estimate(data = example_data, control = slcaControl(em.tol = 1e-6))
summary(model_jlca)
param(model_jlca)

# JLCPA
```

```
nlsy_jlcpa <- slca(
   SMK_98(2) ~ ESMK_98 + FSMK_98 + DSMK_98 + HSMK_98,
   DRK_98(2) ~ EDRK_98 + CDRK_98 + WDRK_98 + BDRK_98,
   MRJ_98(2) ~ EMRJ_98 + CMRJ_98 + OMRJ_98 + SMRJ_98,
   SUB_98(2) ~ SMK_98 + DRK_98 + MRJ_98,
   SMK_03(2) ~ ESMK_03 + FSMK_03 + DSMK_03 + HSMK_03,
   DRK_03(2) ~ EDRK_03 + CDRK_03 + WDRK_03 + BDRK_03,
   MRJ_03(2) ~ EMRJ_03 + CMRJ_03 + OMRJ_03 + SMRJ_03,
   SUB_03(2) ~ SMK_03 + DRK_03 + MRJ_03,
   SMK_08(2) ~ ESMK_08 + FSMK_08 + DSMK_08 + HSMK_08,
   DRK_08(2) ~ EDRK_08 + CDRK_08 + WDRK_08 + BDRK_08,
   MRJ_08(2) ~ EMRJ_08 + CMRJ_08 + OMRJ_08 + SMRJ_08,
   SUB_08(2) ~ SMK_08 + DRK_08 + MRJ_08,
   PROF(2) ~ SUB_98 + SUB_03 + SUB_08,
   constraints = list(
      c("SMK_98", "SMK_03", "SMK_08"),
      c("DRK_98", "DRK_03", "DRK_08"),
      c("MRJ_98", "MRJ_03", "MRJ_08"),
      c("SUB_98 ~ SMK_98", "SUB_03 ~ SMK_03", "SUB_08 ~ SMK_08"),
      c("SUB_98 ~ DRK_98", "SUB_03 ~ DRK_03", "SUB_08 ~ DRK_08"),
      c("SUB_98 ~ MRJ_98", "SUB_03 ~ MRJ_03", "SUB_08 ~ MRJ_08")
   )
) %>% estimate(example_data, control = slcaControl(em.tol = 1e-6))

nlsy_jlcpa$convergence
```

`param` *Print Estimated Parameters of an* `slcafit` *Object*

Description

Prints the estimated parameters of an `slca` model using an `slcafit` object.

Usage

```
param(object, ...)

## S3 method for class 'slcafit'
param(
   object, type = c("probs", "logit"),
   se = FALSE, index = FALSE, ...
)
```

Arguments

`object`	an object of class `slcafit`.
`...`	additional arguments passed to other methods.

type	a character string specifying the format in which the estimated parameters should be displayed. The options are "probs" for probability format or "logit" for log-odds (logit) format. The default setting is "probs".
se	a logical value indicating whether to display standard errors (TRUE) or parameter estimates (FALSE). The default is FALSE. A warning is issued if the information matrix for free parameters is numerically rank deficient. Existing generalized-inverse estimates are retained, but their standard errors and Wald inference may be unreliable.
index	a logical value indicating whether to include (TRUE) or exclude (FALSE) the indices of the estimated parameters in the output. The default is FALSE.

Value

A list containing the requested estimated parameters or their standard errors (if se = TRUE). The components of the list include:

pi	Membership probabilities for the root latent variable.
tau	Conditional probabilities between latent class variables, represented with uppercase letters to account for measurement invariance.
rho	Item response probabilities for each measurement model, represented with lowercase letters to account for measurement invariance.

---

predict.slcafit *Model Predictions for Estimated* slca *Object*

Description

Provides predicted class memberships or posterior probabilities based on a fitted slca model. If newdata is omitted, predictions are computed from the posterior probabilities stored in object.

Usage

```
## S3 method for class 'slcafit'
predict(object, newdata, type = c("class", "posterior"), ...)
```

Arguments

object	An object of class slcafit, representing a fitted slca model.
newdata	A data.frame, matrix, or vector containing the same manifest variables as those used to estimate the object. Unknown response levels are not allowed.
type	A character string indicating the type of prediction. Use "class" to obtain the predicted class membership for each observation and latent class variable, or "posterior" to retrieve posterior probabilities for each class. The default is "class".
...	Additional arguments passed to other methods.

Value

A data.frame or list depending on the type:

- For type = "class", a data.frame is returned where rows represent observations and columns correspond to latent class variables.
- For type = "posterior", a list is returned containing one matrix for each latent class variable. Rows represent observations and columns represent latent classes.

---

regress *Regress Exogenous Variables on Latent Variables*

Description

Performs regression analysis to examine the influence of exogenous (external) variables on latent class variables in an estimated slca model. The function uses logistic regression with a three-step approach to account for measurement error.

Usage

```
regress(object, ...)

## S3 method for class 'slcafit'
regress(
   object, formula, data = parent.frame(),
   imputation = c("modal", "prop"),
   method = c("naive", "BCH", "ML"), ...,
   align = c("rownames", "position")
)

## S3 method for class 'slcafit'
regress(
  object,
  formula,
  data = parent.frame(),
  imputation = c("modal", "prop"),
  method = c("naive", "BCH", "ML"),
  ...,
  align = c("rownames", "position")
)
```

Arguments

object	an object of class slcafit.
...	additional arguments.
formula	a formula specifying the regression model. The left-hand side must be a single latent class variable from the estimated model.

`data` an optional `data.frame` containing the exogenous variables of interest. Variables absent from the fitted data and `data` are taken from the formula's environment and aligned before formula transformations are evaluated. Such vectors and factors need observation IDs in `names()`; matrices and data frames need observation IDs in `rownames()`.

`imputation` a character string specifying the imputation method for latent class assignment. Options include:

- `"modal"`: Assigns each individual to the latent class with the highest posterior probability.
- `"prop"`: Uses posterior probabilities as proportional assignment weights.

`method` a character string specifying the method to adjust for bias in the three-step approach. Options include:

- `"naive"`: A simple approach without correction for classification error.
- `"BCH"`: The bias-adjusted Bolck, Croon, and Hagenaars method.
- `"ML"`: A maximum likelihood approach that accounts for classification error.

`align` how supplied covariate rows are aligned with the fitted data. The default, `"rownames"`, requires unique, nonempty row names and matches them to the fitted observations, regardless of row count or order. Extra rows are allowed, but every fitted observation must be present. Automatic row numbers work only if they still identify the original observations; do not reset them after subsetting or reordering. Use `"position"` explicitly for data already in fitted-observation order; it requires exactly the same number of rows and ignores row names. There is no automatic positional fallback. These rules also apply separately to each environment covariate, even when its length already equals the fitted sample size. Unnamed environment covariates require explicit `align = "position"` and fitted-row order. After observations have been excluded, an unnamed original-length vector cannot be aligned safely: supply `data` with preserved row IDs, or name the vector with its original observation IDs. Matrix covariates remain matrices and factor levels are retained. Unnamed scalar formula parameters are left unchanged, not recycled as covariates. Non-scalar transformation settings (such as spline knots or factor levels) must be written inline or captured inside a transformation function; otherwise they are treated as row inputs. Transformation functions must use their explicit covariate arguments rather than retrieve hidden observation vectors. Use explicit covariate columns instead of dynamic `get()` or `eval()` lookups with row-ID alignment.

Value

A `list` of class `reg.slca` with the following components:

`coefficients` A matrix of regression coefficients representing the odds ratios for each latent class against the baseline class (the last class).

`std.err` A matrix of standard errors corresponding to the regression coefficients.

`vcov` The variance-covariance matrix of the regression coefficients.

`dim` The dimensions of the coefficients matrix.

ll The log-likelihood of the regression model.

The summary function can be used to display the regression coefficients, standard errors, Wald statistics, and p-values. The standard errors are derived by numerically calculated Hessian matrix from nlm function. A warning is issued if the regression information matrix is numerically rank deficient. Existing generalized-inverse estimates are retained, but their standard errors and Wald inference may be unreliable.

References

Vermunt, J. K. (2010). Latent Class Modeling with Covariates: Two Improved Three-Step Approaches. Political Analysis, 18(4), 450–469. http://www.jstor.org/stable/25792024

Examples

```
library(magrittr)
names(nlsy97)
nlsy_jlcpa %>% regress(SMK_98 ~ SEX, nlsy97)

nlsy_jlcpa %>% regress(PROF ~ SEX, nlsy97)
```

---

reorder.slcafit *Reorder Latent Class Membership of Latent Class Variables*

Description

Reorders the latent class membership for specified latent class variables in an slcafit object.

Usage

```
## S3 method for class 'slcafit'
reorder(x, ...)
```

Arguments

x	an object of class slcafit.
...	additional arguments specifying the new order for the latent class variables.

Value

A modified slcafit object with the latent classes reordered according to the specified order.

Examples

```
library(magrittr)
nlsy_jlcpa %>% param

# Reorder the RHO parameters as ascending order
reordered1 <- nlsy_jlcpa %>%
   reorder(SMK_98 = c(1, 3, 2),
           DRK_98 = c(3, 2, 1),
           MRJ_98 = c(3, 1, 2))
reordered1 %>% param
# Label class1: nonuse
#       class2: lifetime use
#       class3: current use

# Reorder the TAU parameters for joint classes as ascending order
reordered2 <- reordered1 %>%
   reorder(SUB_98 = c(3, 4, 5, 1, 2))
reordered2 %>% param
# Label class1: nonuse
#       class2: heavy drinking only
#       class3: not heavy use
#       class4: heavy drinking & smoking
#       class5: heavy use

# Reorder the TAU paramters for profiles as ascending order
reordered3 <- reordered2 %>%
   reorder(PROF = c(4, 1, 3, 2))
reordered3 %>% param
# Label class1: nonuse stayer
#       class2: heavy drinking advancer
#       class3: heavy drk & smk advancer
#       class4: heavy use advancer
```

---

simulate.slca *Simulate Data from an* slca *Model*

Description

Simulates data based on a specified slca model. If the model parameters are not already estimated, they can either be provided by the user or generated randomly.

Usage

```
## S3 method for class 'slca'
simulate(object, nsim = 500, seed = NULL, parm, nlevel, ...)
```

Arguments

object	an slca object representing the model from which data will be simulated.
nsim	a positive number specifying the number of response observations to simulate. Non-integer values are rounded up. The default is 500.
seed	an integer specifying the random seed for reproducibility. If not provided, results will vary across runs.
parm	a user-specified set of parameters to guide the simulation. This is required if the model has not been previously estimated. If the supplied length does not match the model, random parameters are used with a warning.
nlevel	an integer or integer vector specifying the number of levels for each manifest item in the model. If a single integer is provided, all manifest items use that number of levels. Named vectors may specify a subset of manifest items, with unspecified items using the default of 2.
...	Additional arguments passed to other methods.

Value

A list with the following components:

class	A data.frame containing the assigned latent class for each individual across all latent class variables.
response	A data.frame containing the simulated manifest item responses as factors. Internal response encodings used for model calculation are not stored on this object.

Examples

```
m1 <- slca(lc1[3] ~ x1 + x2 + x3 + x4 + x5,
              lc2[4] ~ y1 + y2 + y3 + y4 + y5)
sim <- simulate(m1, 1000)
sapply(sim$class, table)

# simulate data with defined number of levels of manifest items
m2 <- slca(lc1[3] ~ x1 + x2 + x3 + x4)
sim <- simulate(m2, nlevel = c(3, 3, 3, 3))
d <- sim$response
sapply(d, table)

sim <- simulate(m2, nlevel = c(x1 = 2, x3 = 3, x4 = 4))
d <- sim$response
sapply(d, table)

# simulate data with user-defined parameters
pi <- rep(1 / 3, 3)
rho <- c(.9, .1, .9, .1, .9, .1, .9, .1,
         .9, .1, .9, .1, .1, .9, .1, .9,
         .1, .9, .1, .9, .1, .9, .1, .9)
par <- c(pi, rho)
m3 <- slca(lc[3] ~ y1 + y2 + y3 + y4)
```

```
sim <- simulate(m3, parm = par)
mf <- estimate(m3, sim$response)
param(mf)
```

`slca` *Construct Structural Latent Class Model*

Description

Constructs a latent structure with multiple latent class variables.

Usage

```
slca(formula = NULL, ..., constraints = NULL)
```

Arguments

`formula`	a formula specifying the latent structure. Detailed model specifications are provided under 'Details'.
`...`	additional formulae for defining the model structure.
`constraints`	a list of constraints to enforce measurement invariance. Detailed explanations of applying constraints are available under 'Details'.

Details

The `formula` can be categorized into three types, each serving a distinct purpose:

1. **Defining Latent Class Variables with Manifest Indicators**: Specify the relationship between a latent class variable and its manifest indicators. The latent class variable is on the left-hand side (lhs), denoted with square brackets `[]` or parentheses `()` to indicate the number of classes, and manifest indicators are listed on the right-hand side (rhs). For example:

   ```
   LC1[k] ~ x1 + x2 + x3
   LC2[k] ~ y1 + y2 + y3
   LC3(k) ~ z1 + z2 + z3
   ```

   Here, `k` denotes the number of latent classes for the variable.

2. **Relating Latent Class Variables to Each Other**: Define relationships where one latent class variable influences another. For example:

   ```
   LC2 ~ LC1
   ```

   This formula implies that `LC2` is conditionally dependent on `LC1`.

3. **Defining Higher-Level Latent Class Variables**: Specify relationships where a latent class variable is measured by other latent class variables instead of manifest indicators. For example:

```
P[k] ~ LC1 + LC2 + LC3
```

This indicates that the latent variable `P` is measured by the latent class variables `LC1`, `LC2`, and `LC3`.

In all formulas, variables on the lhs influence those on the rhs.

The `constraints` argument enforces specific conditions to ensure precise inference, such as measurement invariance. This is particularly useful for longitudinal analysis (eg. LTA or LCPA), where consistent meanings of latent classes across time are essential.

1. **Measurement Invariance for the Measurement Model**: Ensures probabilities associated with latent class variables remain consistent. For example:

   ```
   c("LC1", "LC2", "LC3")
   ```

   This ensures that `LC1`, `LC2`, and `LC3` have semantically consistent measurement probabilities.

' 2. **Measurement Invariance for the Structural Model**: Applies constraints to ensure consistent interpretations of transition probabilities between latent class variables. For example:

```
c("P ~ LC1", "P -> LC2")
```

This ensures that the transitions from `P` to `LC1` and `P` to `LC2` are consistent.

Value

An object of class `slca` with the following components:

`tree`	A `data.frame` describing the parent-child relationships among latent class and manifest variables.
`latent`	A `data.frame` listing all latent class variables with details such as the number of classes.
`measure`	A `data.frame` describing the measurement model.
`struct`	A `data.frame` detailing the structural model.

The printed model description is divided into four parts:

1. **Latent variables**: Lists the latent class variables and the number of classes for each variable. The root variable is marked with an asterisk (`*`).
2. **Measurement model**: Displays manifest indicators for each latent class variable and any applied measurement constraints (lowercase letters indicate consistency).
3. **Structural model**: Describes the conditional relationships between latent class variables.
4. **Dependency constraints**: Outlines constraints applied to conditional dependencies, where uppercase letters represent consistent dependency structures.

References

Kim, Y. and Chung, H. (2026). slca: An R Package for Structural Modeling for Multiple Latent Class Variables. *Journal of Statistical Software*, **117**(8), 1–52. doi:10.18637/jss.v117.i08

Examples

```
# Standard LCA
slca(lc[3] ~ y1 + y2 + y3 + y4)
# Latent transition analysis (LTA)
slca(lx[3] ~ x1 + x2 + x3 + x4,
     ly[2] ~ y1 + y2 + y3 + y4,
     lx ~ ly)
# LTA with measurement invariance
slca(l1[3] ~ y11 + y21 + y31 + y41,
     l2[3] ~ y12 + y22 + y32 + y42,
     l1 ~ l2, constraints = c("l1", "l2"))
# Joint latent class analysis
slca(lx[2] ~ x1 + x2 + x3 + x4,
     ly[3] ~ y1 + y2 + y3 + y4,
     lz[2] ~ z1 + z2 + z3 + z4,
     jc[3] ~ lx + ly + lz)
# Latent class profile analysis (with measurement invariance)
slca(l1[3] ~ x1 + x2 + x3 + x4,
     l2[3] ~ y1 + y2 + y3 + y4,
     l3[3] ~ z1 + z2 + z3 + z4,
     pf[4] ~ l1 + l2 + l3,
     constraints = c("l1", "l2", "l3"))
```

---

slcaControl *Control Parameters for* slca *Estimation*

Description

Specifies control parameters for estimating slca model.

Usage

```
slcaControl(
  em.iterlim = 5000,
  em.tol = 1e-08,
  nlm.iterlim = 1000,
  nlm.tol = 1e-10,
  init.param = NULL,
  nrep = 1,
  test.iter = 500,
  hessian = FALSE,
  na.rm = FALSE,
  verbose = FALSE
)
```

Arguments

em.iterlim	an integer specifying the maximum number of iterations allowed for the EM algorithm. The default is 5000.

em.tol	a numeric value setting the tolerance for convergence of the EM algorithm. The default is 1e-8.
nlm.iterlim	an integer specifying the maximum number of iterations allowed when using the nlm function for estimation. The default is 1000.
nlm.tol	a numeric value setting the tolerance for convergence of the nlm function. The default is 1e-10.
init.param	a numeric vector specifying the initial parameter values for estimation. Its length must match the number of model parameters.
nrep	an integer specifying the number of estimation trials. The default is 1. Details for generating initial parameter set is described below.
test.iter	an integer specifying the maximum number of iterations allowed for parameter testing. The default is 500.
hessian	a logical value indicating whether to calculate Hessian via nlm function numerically, if so, vcov method can provide variance-covariance matrix with Hessian instead of outer-product-of-gradients (OPG). The default is FALSE.
na.rm	a logical value indicating whether to remove observations containing missing values (NA). The default is FALSE. Details for treating missing data is described below.
verbose	a logical value indicating whether to display progress updates during the estimation process. The default is FALSE.
	A list containing control parameters for slca estimation, including the specified iteration limits, tolerances, and additional options.

Details

Missing data: Missing data are handled in two ways. If all manifest variables for an observation are missing, the case is excluded (listwise deletion). For partially missing data, the model assumes Missing At Random (MAR) and uses following algorithm to integrate over the missing values. Saturated-model fit statistics use response cells compatible with the observed patterns. If the required table is too large to construct, or its EM iteration does not converge within the iteration limit, these statistics are returned as NA with a warning; model estimation remains available. The fitted model frame records saturated.converged and saturated.iterations attributes for this calculation. In the E-step, posterior probabilities are computed using only the observed items. In the M-step, parameter updates use these posterior probabilities, with expected counts for missing responses distributed according to current parameter estimates.

Local maxima and multiple starting values: The EM algorithm may converge to a local rather than a global maximum. To reduce this risk, the nrep option allows multiple repetitions with different initial values. PI parameters are initialized equally (e.g., 0.5 for two classes), while TAU and RHO parameters are given small random perturbations to uniform probabilities, normalized within each parent state. The test.iter option runs a limited number of iterations for each initial set, and the best-performing set is then used for full convergence, improving the chance of reaching the global maximum.

See Also

slca

Index

∗ **datasets**
addhealth, 2
gss7677, 11
nlsy97, 13
nlsy_jlcpa, 12

addhealth, 2

compare, 4, *10*
confint.slcafit, 6

estimate, 7

gof, *5*, *8*, 9
gss7677, 11

nlsy97, 13
nlsy_jlcpa, 12
nlsy_jlta *(*nlsy_jlcpa*)*, 12

param, *7, 8*, 15
param(), *8*
predict.slcafit, *8*, 16

regress, *8*, 17
reorder.slcafit, *8*, *13*, 19

simulate.slca, 20
slca, 22, *25*
slca(), *8*
slcaControl, 24
slcaControl(), *7, 8*