Package 'regsem'

January 3, 2019

Type Package

Title Regularized Structural Equation Modeling

Version 1.2.3

Author Ross Jacobucci[aut,cre],
Kevin J. Grimm [ctb],
Andreas M. Brandmaier [ctb],
Sarfaraz Serang [ctb],
Rogier A. Kievit [ctb]

Maintainer Ross Jacobucci <rcjacobuc@gmail.com>

Description Uses both ridge and lasso penalties (and extensions) to penalize specific parameters in structural equation models. The package offers additional cost functions, cross validation, and other extensions beyond traditional structural equation models. Also contains a function to perform exploratory mediation (XMed).

License GPL (>= 2)

LazyData TRUE

VignetteBuilder knitr

Depends lavaan, Rcpp, Rsolnp

Suggests snowfall, MASS, GA, caret, glmnet, ISLR, lbfgs, numDeriv, psych, knitr, nloptr, NlcOptim, optimx, semPlot, colorspace

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.1.0

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-01-02 23:10:18 UTC

R topics documented:

`cv_regsem` *The main function that runs multiple penalty values.*

Description

The main function that runs multiple penalty values.

Usage

```
cv_regsem(model, n.lambda = 40, pars_pen = "regressions",
  metric = ifelse(fit.ret2 == "train", "BIC", "chisq"),
  mult.start = FALSE, multi.iter = 10, jump = 0.01,
  lambda.start = 0, alpha = 0.5, gamma = 3.7, type = "lasso",
  fit.ret = c("rmsea", "BIC", "chisq"), fit.ret2 = "train",
  n.boot = 20, data = NULL, optMethod = "rsolnp", gradFun = "ram",
  hessFun = "none", test.cov = NULL, prerun = FALSE,
  parallel = FALSE, ncore = 2, Start = "lavaan", subOpt = "nlminb",
  diff_par = NULL, LB = -Inf, UB = Inf, par.lim = c(-Inf, Inf),
  block = TRUE, full = TRUE, calc = "normal", max.iter = 2000,
  tol = 1e-05, solver = FALSE, quasi = FALSE, solver.maxit = 5,
  alpha.inc = FALSE, step = 0.1, momentum = FALSE,
  step.ratio = FALSE, line.search = FALSE, nlminb.control = list(),
  warm.start = FALSE, missing = "listwise", verbose = TRUE, ...)
```

Arguments

`model`	Lavaan output object. This is a model that was previously run with any of the lavaan main functions: cfa(), lavaan(), sem(), or growth(). It also can be from the efaUnrotate() function from the semTools package. Currently, the parts of the model which cannot be handled in regsem is the use of multiple group models, missing other than listwise, thresholds from categorical variable models, the use of additional estimators other than ML, most notably WLSMV for categorical variables. Note: the model does not have to actually run (use do.fit=FALSE), converge etc... regsem() uses the lavaan object as more of a parser and to get sample covariance matrix.

n.lambda	number of penalization values to test.
pars_pen	Parameter indicators to penalize. There are multiple ways to specify. The default is to penalize all regression parameters ("regressions"). Additionally, one can specify all loadings ("loadings"), or both c("regressions","loadings"). Next, parameter labels can be assigned in the lavaan syntax and passed to pars_pen. See the example.Finally, one can take the parameter numbers from the A or S matrices and pass these directly. See extractMatrices(lav.object)$A.
metric	Which fit index to use to choose a final model? Note that it chooses the best fit that also achieves convergence (conv=0).
mult.start	Logical. Whether to use multi_optim() (TRUE) or regsem() (FALSE).
multi.iter	maximum number of random starts for multi_optim
jump	Amount to increase penalization each iteration.
lambda.start	What value to start the penalty at
alpha	Mixture for elastic net. 1 = ridge, 0 = lasso
gamma	Additional penalty for MCP and SCAD
type	Penalty type. Options include "none", "lasso", "ridge", "enet" for the elastic net, "alasso" for the adaptive lasso and "diff_lasso". diff_lasso penalizes the discrepency between parameter estimates and some pre-specified values. The values to take the deviation from are specified in diff_par. Two methods for sparser results than lasso are the smooth clipped absolute deviation, "scad", and the minimum concave penalty, "mcp".
fit.ret	Fit indices to return.
fit.ret2	Return fits using only dataset "train" or bootstrap "boot"? Have to do 2 sample CV manually.
n.boot	Number of bootstrap samples if fit.ret2="boot"
data	Optional dataframe. Only required for missing="fiml".
optMethod	Solver to use. Two main options for use: rsoolnp and coord_desc. Although slightly slower, rsolnp works much better for complex models. coord_desc uses gradient descent with soft thresholding for the type of of penalty. Rsolnp is a nonlinear solver that doesn't rely on gradient information. There is a similar type of solver also available for use, slsqp from the nloptr package. coord_desc can also be used with hessian information, either through the use of quasi=TRUE, or specifying a hess_fun. However, this option is not recommended at this time.
gradFun	Gradient function to use. Recommended to use "ram", which refers to the method specified in von Oertzen & Brick (2014). Only for use with optMethod="coord_desc".
hessFun	hessian function to use. Currently not recommended.
test.cov	Covariance matrix from test dataset. Necessary for CV=T
prerun	Logical. Use rsolnp to first optimize before passing to gradient descent? Only for use with coord_desc
parallel	Logical. whether to parallelize the processes running models for all values of lambda.
ncore	Number of cores to use when parallel=TRUE

Start	type of starting values to use.
subOpt	type of optimization to use in the optimx package.
diff_par	parameter values to deviate from.
LB	lower bound vector.
UB	upper bound vector
par.lim	Vector of minimum and maximum parameter estimates. Used to stop optimization and move to new starting values if violated.
block	Whether to use block coordinate descent
full	Whether to do full gradient descent or block
calc	Type of calc function to use with means or not. Not recommended for use.
max.iter	Number of iterations for coordinate descent
tol	Tolerance for coordinate descent
solver	Whether to use solver for coord_desc
quasi	Whether to use quasi-Newton
solver.maxit	Max iterations for solver in coord_desc
alpha.inc	Whether alpha should increase for coord_desc
step	Step size
momentum	Momentum for step sizes
step.ratio	Ratio of step size between A and S. Logical
line.search	Use line search for optimization. Default is no, use fixed step size
nlminb.control	list of control values to pass to nlminb
warm.start	Whether start values are based on previous iteration. This is not recommended.
missing	How to handle missing data. Current options are "listwise" and "fiml".
verbose	Print progress bar?
...	Any additional arguments to pass to regsem() or multi_optim().

Examples

```
## Not run:
library(regsem)
vignette("overview",package="regsem")
# put variables on same scale for regsem
HS <- data.frame(scale(HolzingerSwineford1939[,7:15]))
mod <- '
f =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
'
outt = cfa(mod, HS)
# increase to > 25
cv.out = cv_regsem(outt,type="lasso", pars_pen=c(1:2,6:8),
          n.lambda=5,jump=0.01)
# check parameter numbers
extractMatrices(outt)["A"]
# equivalent to
```

```
mod <- '
f =~ 1*x1 + l1*x2 + l2*x3 + l3*x4 + l4*x5 + l5*x6 + l6*x7 + l7*x8 + l8*x9
'
outt = cfa(mod,HS)
# increase to > 25
cv.out = cv_regsem(outt, type="lasso", pars_pen=c("l1","l2","l6","l7","l8"),
         n.lambda=5,jump=0.01)
summary(cv.out)
plot(cv.out, show.minimum="BIC")

mod <- '
f =~ x1 + x2 + x3 + x4 + x5 + x6
'
outt = cfa(mod, HS)
# can penalize all loadings
cv.out = cv_regsem(outt,type="lasso", pars_pen="loadings",
                   n.lambda=5,jump=0.01)

mod2 <- '
f =~ x4+x5+x3
#x1 ~ x7 + x8 + x9 + x2
x1 ~ f
x2 ~ f
'
outt2 = cfa(mod2, HS)
extractMatrices(outt2)$A
# if no pars_pen specification, defaults to all
# regressions
cv.out = cv_regsem(outt2,type="lasso",
                   n.lambda=15,jump=0.03)
# check
cv.out$pars_pen

## End(Not run)
```

`extractMatrices` *This function extracts RAM matrices from a lavaan object.*

Description

This function extracts RAM matrices from a lavaan object.

Usage

```
extractMatrices(model)
```

Arguments

`model`	Lavaan model object.

Value

The RAM matrices from `model`.

Examples

```
library(lavaan)
data(HolzingerSwineford1939)
HS.model <- ' visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9 '
mod <- cfa(HS.model, data=HolzingerSwineford1939)
mats = extractMatrices(mod)
```

---

fit_indices *Calculates the fit indices*

Description

Calculates the fit indices

Usage

```
fit_indices(model, CV = F, CovMat = NULL, data = NULL,
  n.obs = NULL)
```

Arguments

`model`	regsem model object.
`CV`	cross-validation. Note that this requires splitting the dataset into a training and test set prior to running the model. The model should be run on the training set, with the test set held out and then passed to CovMat=.
`CovMat`	If CV=T then test covariance matrix must be supplied. Note That this should be done before running the lavaan model and should not overlap with the data or covariance matrix used to run the model.
`data`	supply the dataset?
`n.obs`	Number of observations in the test set for CV.

Examples

```
## Not run:
fit_indices()

## End(Not run)
```

multi_optim *Multiple starts for Regularized Structural Equation Modeling*

Description

Multiple starts for Regularized Structural Equation Modeling

Usage

```
multi_optim(model, max.try = 10, lambda = 0, alpha = 0.5,
  gamma = 3.7, LB = -Inf, UB = Inf, par.lim = c(-Inf, Inf),
  block = TRUE, full = TRUE, type = "lasso", optMethod = "rsolnp",
  gradFun = "ram", pars_pen = "regressions", diff_par = NULL,
  hessFun = "none", tol = 1e-05, solver = FALSE, quasi = FALSE,
  solver.maxit = 50000, alpha.inc = FALSE, line.search = FALSE,
  prerun = FALSE, step = 0.1, momentum = FALSE, step.ratio = FALSE,
  verbose = FALSE, warm.start = FALSE, Start2 = NULL,
  nlminb.control = NULL, max.iter = 500)
```

Arguments

model	Lavaan output object. This is a model that was previously run with any of the lavaan main functions: cfa(), lavaan(), sem(), or growth(). It also can be from the efaUnrotate() function from the semTools package. Currently, the parts of the model which cannot be handled in regsem is the use of multiple group models, missing other than listwise, thresholds from categorical variable models, the use of additional estimators other than ML, most notably WLSMV for categorical variables. Note: the model does not have to actually run (use do.fit=FALSE), converge etc... regsem() uses the lavaan object as more of a parser and to get sample covariance matrix.
max.try	number of starts to try before convergence.
lambda	Penalty value. Note: higher values will result in additional convergence issues.
alpha	Mixture for elastic net.
gamma	Additional penalty for MCP and SCAD
LB	lower bound vector. Note: This is very important to specify when using regularization. It greatly increases the chances of converging.
UB	Upper bound vector
par.lim	Vector of minimum and maximum parameter estimates. Used to stop optimization and move to new starting values if violated.
block	Whether to use block coordinate descent
full	Whether to do full gradient descent or block
type	Penalty type. Options include "none", "lasso", "enet" for the elastic net, "alasso" for the adaptive lasso and "diff_lasso". If ridge penalties are desired, use type="enet" and alpha=1. diff_lasso penalizes the discrepency between parameter estimates

	and some pre-specified values. The values to take the deviation from are specified in diff_par. Two methods for sparser results than lasso are the smooth clipped absolute deviation, "scad", and the minimum concave penalty, "mcp".
optMethod	Solver to use. Two main options for use: rsoolnp and coord_desc. Although slightly slower, rsolnp works much better for complex models. coord_desc uses gradient descent with soft thresholding for the type of of penalty. Rsolnp is a nonlinear solver that doesn't rely on gradient information. There is a similar type of solver also available for use, slsqp from the nloptr package. coord_desc can also be used with hessian information, either through the use of quasi=TRUE, or specifying a hess_fun. However, this option is not recommended at this time.
gradFun	Gradient function to use. Recommended to use "ram", which refers to the method specified in von Oertzen & Brick (2014). Only for use with optMethod="coord_desc".
pars_pen	Parameter indicators to penalize. There are multiple ways to specify. The default is to penalize all regression parameters ("regressions"). Additionally, one can specify all loadings ("loadings"), or both c("regressions","loadings"). Next, parameter labels can be assigned in the lavaan syntax and passed to pars_pen. See the example.Finally, one can take the parameter numbers from the A or S matrices and pass these directly. See extractMatrices(lav.object)$A.
diff_par	Parameter values to deviate from. Only used when type="diff_lasso".
hessFun	Hessian function to use. Currently not recommended.
tol	Tolerance for coordinate descent
solver	Whether to use solver for coord_desc
quasi	Whether to use quasi-Newton. Currently not recommended.
solver.maxit	Max iterations for solver in coord_desc
alpha.inc	Whether alpha should increase for coord_desc
line.search	Use line search for optimization. Default is no, use fixed step size
prerun	Logical. Use rsolnp to first optimize before passing to gradient descent? Only for use with coord_desc.
step	Step size
momentum	Momentum for step sizes
step.ratio	Ratio of step size between A and S. Logical
verbose	Whether to print iteration number.
warm.start	Whether start values are based on previous iteration. This is not recommended.
Start2	Provided starting values. Not required
nlminb.control	list of control values to pass to nlminb
max.iter	Number of iterations for coordinate descent

Examples

```
## Not run:
# Note that this is not currently recommended. Use cv_regsem() instead
library(regsem)
# put variables on same scale for regsem
```

```
HS <- data.frame(scale(HolzingerSwineford1939[ ,7:15]))
mod <- '
f =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
'
outt = cfa(mod, HS, meanstructure=TRUE)

fit1 <- multi_optim(outt, max.try=40,
                    lambda=0.1, type="lasso")


# growth model
model <- ' i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
          s =~ 0*t1 + s1*t2 + s2*t3 + 3*t4 '
fit <- growth(model, data=Demo.growth)
summary(fit)
fitmeasures(fit)
fit3 <- multi_optim(fit, lambda=0.2, type="lasso")
summary(fit3)

## End(Not run)
```

`parse_parameters` *Takes either a vector of parameter ids or a vector of named parameters and returns a vector of parameter ids*

Description

Takes either a vector of parameter ids or a vector of named parameters and returns a vector of parameter ids

Usage

```
parse_parameters(x, model)
```

Arguments

`x`	Parameter labels
`model`	Lavaan model

Value

NULL if undefined input. Else vector of parameter ids

plot.cvregsem *Plot function for cv_regsem*

Description

Plot function for cv_regsem

Usage

```
## S3 method for class 'cvregsem'
plot(x, ..., pars = NULL, show.minimum = "BIC",
  col = NULL, type = "l", lwd = 3, h_line = 0, lty = 1,
  xlab = NULL, ylab = NULL, legend.x = NULL, legend.y = NULL,
  legend.cex = 1, legend.bg = par("bg"), grey.out = FALSE)
```

Arguments

x	An x from cv_regsem.
...	Other arguments.
pars	Which parameters to plot
show.minimum	What fit index to use
col	A specification for the default plotting color.
type	what type of plot should be drawn. Possible types are "p" for points, "l" for lines, or "b" for both
lwd	line width
h_line	Where to draw horizontal line
lty	line type
xlab	X axis label
ylab	Y axis label
legend.x	x-coordinate of legend. See ?legend
legend.y	y-coordinate of legend. See ?legend
legend.cex	cex of legend. See ?legend
legend.bg	legend background color. See ?legend
grey.out	Add grey to background

rcpp_fit_fun *Calculates the objective function values.*

Description

Calculates the objective function values.

Usage

```
rcpp_fit_fun(ImpCov, SampCov, type2, lambda, gamma, pen_vec, pen_diff,
  e_alpha)
```

Arguments

ImpCov	expected covariance matrix.
SampCov	Sample covariance matrix.
type2	penalty type.
lambda	penalty value.
gamma	additional penalty for mcp and scad
pen_vec	vector of penalized parameters.
pen_diff	Vector of values to take deviation from.
e_alpha	Alpha for elastic net

rcpp_grad_ram *Calculates the gradient vector based on Von Oertzen \& Brick, 2014*

Description

Calculates the gradient vector based on Von Oertzen \& Brick, 2014

Usage

```
rcpp_grad_ram(par, ImpCov, SampCov, Areg, Sreg, A, S, F, lambda, type2,
  pen_vec, diff_par)
```

Arguments

par	vector with parameters.
ImpCov	expected covariance matrix.
SampCov	Sample covariance matrix.
Areg	A matrix with current parameter estimates.
Sreg	S matrix with current parameter estimates.

A	A matrix with parameter labels.
S	S matrix with parameter labels.
F	F matrix.
lambda	penalty value.
type2	penalty type.
pen_vec	parameter indicators to be penalized.
diff_par	parameter values to take deviations from.

`rcpp_quasi_calc` *Compute quasi Hessian*

Description

Compute quasi Hessian

Usage

```
rcpp_quasi_calc(I, s, y, H)
```

Arguments

I	identity matrix.
s	s vector.
y	y vector.
H	previous Hessian.

`rcpp_RAMmult` *Take RAM matrices, multiplies, and returns Implied Covariance matrix.*

Description

Take RAM matrices, multiplies, and returns Implied Covariance matrix.

Usage

```
rcpp_RAMmult(par, A, S, S_fixed, A_fixed, A_est, S_est, F, I)
```

Arguments

par	parameter estimates.
A	A matrix with parameter labels.
S	S matrix with parameter labels.
S_fixed	S matrix with fixed indicators.
A_fixed	A matrix with fixed indicators.
A_est	A matrix with parameter estimates.
S_est	S matrix with parameter estimates.
F	F matrix.
I	Diagonal matrix of ones.

---

regsem *Regularized Structural Equation Modeling. Tests a single penalty. For testing multiple penalties, see cv_regsem().*

Description

Regularized Structural Equation Modeling. Tests a single penalty. For testing multiple penalties, see cv_regsem().

Usage

```
regsem(model, lambda = 0, alpha = 0.5, gamma = 3.7, type = "lasso",
  data = NULL, optMethod = "rsolnp", estimator = "ML",
  gradFun = "ram", hessFun = "none", prerun = FALSE,
  parallel = "no", Start = "lavaan", subOpt = "nlminb",
  longMod = F, pars_pen = "regressions", diff_par = NULL,
  LB = -Inf, UB = Inf, par.lim = c(-Inf, Inf), block = TRUE,
  full = TRUE, calc = "normal", max.iter = 500, tol = 1e-05,
  solver = FALSE, quasi = FALSE, solver.maxit = 5,
  alpha.inc = FALSE, line.search = FALSE, step = 0.1,
  momentum = FALSE, step.ratio = FALSE, nlminb.control = list(),
  missing = "listwise")
```

Arguments

model	Lavaan output object. This is a model that was previously run with any of the lavaan main functions: cfa(), lavaan(), sem(), or growth(). It also can be from the efaUnrotate() function from the semTools package. Currently, the parts of the model which cannot be handled in regsem is the use of multiple group models, missing other than listwise, thresholds from categorical variable models, the use of additional estimators other than ML, most notably WLSMV for categorical variables. Note: the model does not have to actually run (use do.fit=FALSE), converge etc... regsem() uses the lavaan object as more of a parser and to get sample covariance matrix.

lambda	Penalty value. Note: higher values will result in additional convergence issues. If using values > 0.1, it is recommended to use mutli_optim() instead. See multi_optim for more detail.
alpha	Mixture for elastic net. 1 = ridge, 0 = lasso
gamma	Additional penalty for MCP and SCAD
type	Penalty type. Options include "none", "lasso", "enet" for the elastic net, "alasso" for the adaptive lasso and "diff_lasso". If ridge penalties are desired, use type="enet" and alpha=1. diff_lasso penalizes the discrepency between parameter estimates and some pre-specified values. The values to take the deviation from are specified in diff_par. Two methods for sparser results than lasso are the smooth clipped absolute deviation, "scad", and the minimum concave penalty, "mcp".
data	Optional dataframe. Only required for missing="fiml" which is not currently working.
optMethod	Solver to use. Two main options for use: rsoolnp and coord_desc. Although slightly slower, rsolnp works much better for complex models. coord_desc uses gradient descent with soft thresholding for the type of of penalty. Rsolnp is a nonlinear solver that doesn't rely on gradient information. There is a similar type of solver also available for use, slsqp from the nloptr package. coord_desc can also be used with hessian information, either through the use of quasi=TRUE, or specifying a hess_fun. However, this option is not recommended at this time.
estimator	Whether to use maximum likelihood (ML) or unweighted least squares (ULS) as a base estimator.
gradFun	Gradient function to use. Recommended to use "ram", which refers to the method specified in von Oertzen & Brick (2014). Only for use with optMethod="coord_desc".
hessFun	Hessian function to use. Recommended to use "ram", which refers to the method specified in von Oertzen & Brick (2014). This is currently not recommended.
prerun	Logical. Use rsolnp to first optimize before passing to gradient descent? Only for use with coord_desc.
parallel	Logical. Whether to parallelize the processes?
Start	type of starting values to use. Only recommended to use "default". This sets factor loadings and variances to 0.5. Start = "lavaan" uses the parameter estimates from the lavaan model object. This is not recommended as it can increase the chances in getting stuck at the previous parameter estimates.
subOpt	Type of optimization to use in the optimx package.
longMod	If TRUE, the model is using longitudinal data? This changes the sample covariance used.
pars_pen	Parameter indicators to penalize. There are multiple ways to specify. The default is to penalize all regression parameters ("regressions"). Additionally, one can specify all loadings ("loadings"), or both c("regressions","loadings"). Next, parameter labels can be assigned in the lavaan syntax and passed to pars_pen. See the example.Finally, one can take the parameter numbers from the A or S matrices and pass these directly. See extractMatrices(lav.object)$A.
diff_par	Parameter values to deviate from. Only used when type="diff_lasso".

LB	lower bound vector. Note: This is very important to specify when using regularization. It greatly increases the chances of converging.
UB	Upper bound vector
par.lim	Vector of minimum and maximum parameter estimates. Used to stop optimization and move to new starting values if violated.
block	Whether to use block coordinate descent
full	Whether to do full gradient descent or block
calc	Type of calc function to use with means or not. Not recommended for use.
max.iter	Number of iterations for coordinate descent
tol	Tolerance for coordinate descent
solver	Whether to use solver for coord_desc
quasi	Whether to use quasi-Newton
solver.maxit	Max iterations for solver in coord_desc
alpha.inc	Whether alpha should increase for coord_desc
line.search	Use line search for optimization. Default is no, use fixed step size
step	Step size
momentum	Momentum for step sizes
step.ratio	Ratio of step size between A and S. Logical
nlminb.control	list of control values to pass to nlminb
missing	How to handle missing data. Current options are "listwise" and "fiml". "fiml" is not currently working well.

Value

out List of return values from optimization program

convergence Convergence status. 0 = converged, 1 or 99 means the model did not converge.

par.ret Final parameter estimates

Imp_Cov Final implied covariance matrix

grad Final gradient.

KKT1 Were final gradient values close enough to 0.

KKT2 Was the final Hessian positive definite.

df Final degrees of freedom. Note that df changes with lasso penalties.

npar Final number of free parameters. Note that this can change with lasso penalties.

SampCov Sample covariance matrix.

fit Final F_ml fit. Note this is the final parameter estimates evaluated with the F_ml fit function.

coefficients Final parameter estimates

nvar Number of variables.

N sample size.

nfac Number of factors

baseline.chisq Baseline chi-square.

baseline.df Baseline degrees of freedom.

Examples

```
# Note that this is not currently recommended. Use cv_regsem() instead
# vignette("overview",package="regsem")
library(lavaan)
# put variables on same scale for regsem
HS <- data.frame(scale(HolzingerSwineford1939[,7:15]))
mod <- '
f =~ 1*x1 + l1*x2 + l2*x3 + l3*x4 + l4*x5 + l5*x6 + l6*x7 + l7*x8 + l8*x9
'
# Recommended to specify meanstructure in lavaan
outt = cfa(mod, HS, meanstructure=TRUE)

fit1 <- regsem(outt, lambda=0.05, type="lasso",
  pars_pen=c("l1", "l2", "l6", "l7", "l8"))
#equivalent to pars_pen=c(1:2, 6:8)
#summary(fit1)
```

---

summary.cvregsem *print information about cvregsem object*

Description

print information about cvregsem object

Usage

```
## S3 method for class 'cvregsem'
summary(object, ...)
```

Arguments

object	cv_regsem object
...	Additional arguments

---

summary.regsem *Summary results from regsem.*

Description

Summary results from regsem.

Usage

```
## S3 method for class 'regsem'
summary(object, ...)
```

Arguments

object	An object from regsem.
...	Other arguments.

---

xmed *Function to performed exploratory mediation with continuous and categorical variables*

Description

Function to performed exploratory mediation with continuous and categorical variables

Usage

```
xmed(data, iv, mediators, dv, covariates = NULL, type = "lasso",
  nfolds = 10, epsilon = 0.001, seed = NULL)
```

Arguments

data	Name of the dataset
iv	Name of independent variable
mediators	Name of mediators
dv	Name of dependent variable
covariates	Name of covariates to be included in model.
type	What type of penalty. Options include lasso, ridge, and enet.
nfolds	Number of cross-validation folds.
epsilon	Threshold for determining whether effect is 0 or not.
seed	Set seed to control CV results

Examples

```
## Not run:
# example
library(ISLR)
College1 = College[which(College$Private=="Yes"),]
Data = data.frame(scale(College1[c("Grad.Rate","Accept","Outstate","Room.Board","Books","Expend")]))
Data$Grad.Rate <- ifelse(Data$Grad.Rate > 0,1,0)
Data$Grad.Rate <- as.factor(Data$Grad.Rate)
#lavaan model with all mediators
model1 <-
 ' # direct effect (c_prime)
Grad.Rate ~ c_prime*Accept
# mediators
Outstate ~ a1*Accept
Room.Board ~ a2*Accept
```

```
Books ~ a3*Accept
Expend ~ a6*Accept
Grad.Rate ~ b1*Outstate + b2*Room.Board + b3*Books + b6*Expend
# indirect effects (a*b)
a1b1 := a1*b1
a2b2 := a2*b2
a3b3 := a3*b3
a6b6 := a6*b6
# total effect (c)
c := c_prime + (a1*b1) + (a2*b2) + (a3*b3) + (a6*b6)
'
#p-value approach using delta method standard errors
fit.delta = sem(model1,data=Data,fixed.x=TRUE,ordered="Grad.Rate")
summary(fit.delta)

#xmed()

iv <- "Accept"
dv <- "Grad.Rate"
mediators <- c("Outstate","Room.Board","Books","Expend")

out <- xmed(Data,iv,mediators,dv)
out

## End(Not run)
```

Index