The sem Package

February 8, 2009

Version 0.9-15

Date 2009/01/26

Title Structural Equation Models

Author John Fox <jfox@mcmaster.ca>

Maintainer John Fox <jfox@mcmaster.ca>

Depends R (>= 1.6.0), stats

Suggests boot

Description This package contains functions for fitting general linear structural equation models (with observed and unobserved variables) by the method of maximum likelihood using the RAM approach, and for fitting structural equations in observed-variable models by two-stage least squares.

License GPL (>= 2)

URL http://www.r-project.org, http://www.socsci.mcmaster.ca/jfox/

Revision 9

R topics documented:

`CNES` *Variables from the 1997 Canadian National Election Study*

Description

These variables are from the mailback questionnaire to the 1997 Canadian National Election Study, and are intended to tap attitude towards "traditional values."

Usage

```
data(CNES)
```

Format

A data frame with 1529 observations on the following 4 variables.

`MBSA2` an ordered factor with levels `StronglyDisagree` `Disagree` `Agree` `StronglyAgree`, in response to the statement, "We should be more tolerant of people who choose to live according to their own standards, even if they are very different from our own."

`MBSA7` an ordered factor with levels `StronglyDisagree` `Disagree` `Agree` `StronglyAgree`, in response to the statement, "Newer lifestyles are contributing to the breakdown of our society."

`MBSA8` an ordered factor with levels `StronglyDisagree` `Disagree` `Agree` `StronglyAgree`, in response to the statement, "The world is always changing and we should adapt our view of moral behaviour to these changes."

`MBSA9` an ordered factor with levels `StronglyDisagree` `Disagree` `Agree` `StronglyAgree`, in response to the statement, "This country would have many fewer problems if there were more emphasis on traditional family values."

Source

York University Institute for Social Research.

`Klein` *Klein's Data on the U. S. Economy*

Description

Data for Klein's (1950) simple econometric model of the U. S. economy.

The `Klein` data frame has 22 rows and 10 columns.

Usage

```
data(Klein)
```

Format

This data frame contains the following columns:

Year 1921–1941

C consumption.

P private profits.

Wp private wages.

I investment.

K.lag capital stock, lagged one year.

X equilibrium demand.

Wg government wages.

G government non-wage spending.

T indirect business taxes and net exports.

Source

Greene, W. H. (1993) *Econometric Analysis, Second Edition.* Macmillan.

References

Klein, L. (1950) *Economic Fluctuations in the United States 1921–1941.* Wiley.

Examples

```
data(Klein)

Klein$P.lag <- c(NA, Klein$P[-22])
Klein$X.lag <- c(NA, Klein$X[-22])

summary(tsls(C ~ P + P.lag + I(Wp + Wg),
    instruments=~1 + G + T + Wg + I(Year - 1931) + K.lag + P.lag + X.lag,
    data=Klein))

summary(tsls(I ~ P + P.lag + K.lag,
    instruments=~1 + G + T + Wg + I(Year - 1931) + K.lag + P.lag + X.lag,
    data=Klein))

summary(tsls(Wp ~ X + X.lag + I(Year - 1931),
    instruments=~1 + G + T + Wg + I(Year - 1931) + K.lag + P.lag + X.lag,
    data=Klein))
```

`Kmenta` *Partly Artificial Data on the U. S. Economy*

Description

These are partly contrived data from Kmenta (1986), constructed to illustrate estimation of a simultaneous-equation model.

The `Kmenta` data frame has 20 rows and 5 columns.

Usage

```
data(Kmenta)
```

Format

This data frame contains the following columns:

Q food consumption per capita.

P ratio of food prices to general consumer prices.

D disposable income in constant dollars.

F ratio of preceding year's prices received by farmers to general consumer prices.

A time in years.

Details

The exogenous variables `D`, `F`, and `A` are based on real data; the endogenous variables `P` and `Q` were generated by simulation.

Source

Kmenta, J. (1986) *Elements of Econometrics, Second Edition*, Macmillan.

Examples

```
data(Kmenta)
```

boot.sem *Bootstrap a Structural Equation Model*

Description

Bootstraps a structural equation model in an sem object (as returned by the sem function).

Usage

```
boot.sem(data, model, R=100, cov=cov, ...)

## S3 method for class 'bootsem':
print(x, digits=getOption("digits"), ...)

## S3 method for class 'bootsem':
summary(object,
    type=c("perc", "bca", "norm", "basic", "none"), level=0.95, ...)
```

Arguments

data	a data frame containing the data to which the model was fit; note that the original observations are required, not just the covariance matrix of the observed variables in the model.
model	an sem object, produced by the sem function.
R	the number of bootstrap replications; the default is 100, which should be enough for computing standard errors, but not confidence intervals (except for the normal-theory intervals).
cov	a function to compute the input covariance matrix; the default is cov. Use cor if the model is fit to the correlation matrix. The function hetcor in the polycor package will compute product-moment, polychoric, and polyserial correlations among mixed continuous and ordinal variables (see the example below for an illustration).
x, object	an object of class bootsem.
digits	controls the number of digits to print.
type	type of bootstrapped confidence intervals to compute; the default is "perc" (percentile); see boot.ci for details.
level	level for confidence intervals; default is 0.95.
...	in boot.sem, arguments to be passed to sem; otherwise ignored.

Details

boot.sem implements the nonparametric bootstrap, assuming an independent random sample. Convergence failures in the bootstrap resamples are discarded (and a warning printed); 10 consecutive convergence failures result in an error. You can use the boot function in the boot package for more complex sampling schemes and additional options.

Bootstrapping is implemented by resampling the observations in `data`, recalculating the input covariance matrix with `cov` and refitting the model with `sem`, using the parameter estimates from the original sample as start-values.

Warning: the bootstrapping process can be very time-consuming.

Value

`boot.sem` returns an object of class `bootsem`, which inherits from class `boot`, supported by the `boot` package. The returned object contains the following components:

`t0`	the estimated parameters in the model fit to the original data set.
`t`	a matrix containing the bootstrapped estimates, one bootstrap replication per row.
`data`	the data frame containing the data to which the model was fit.
`seed`	the value of `.Random.seed` when `boot.sem` was called.
`statistic`	the function used to produce the bootstrap replications; this is always the local function `refit` from `boot.sem`.
`sim`	always set to `"ordinary"`; see the documentation for the `boot` function.
`stype`	always set to `"i"`; see the documentation for the `boot` function.
`call`	the call of the `boot.sem` function.
`strata`	always a vector of ones.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

References

Davison, A. C., and Hinkley, D. V. (1997) *Bootstrap Methods and their Application.* Cambridge.

Efron, B., and Tibshirani, R. J. (1993) *An Introduction to the Bootstrap.* Chapman and Hall.

See Also

`boot`, `sem`

Examples

```
    ## Not run:

# A simple confirmatory factor-analysis model using polychoric correlations.
#  The polycor package is required for the hetcor function.

library(polycor)

# The following function returns correlations computed by hetcor,
#   and is used for the bootstrapping.

hcor <- function(data) hetcor(data, std.err=FALSE)$correlations
```

```
model.cnes <- specify.model()
F -> MBSA2, lam1, NA
F -> MBSA7, lam2, NA
F -> MBSA8, lam3, NA
F -> MBSA9, lam4, NA
F <-> F, NA, 1
MBSA2 <-> MBSA2, the1, NA
MBSA7 <-> MBSA7, the2, NA
MBSA8 <-> MBSA8, the3, NA
MBSA9 <-> MBSA9, the4, NA

data(CNES)
R.cnes <- hcor(CNES)

sem.cnes <- sem(model.cnes, R.cnes, N=1529)
summary(sem.cnes)

#  Note: this can take a couple of minutes:

system.time(boot.cnes <- boot.sem(CNES, sem.cnes, R=100, cov=hcor), gcFirst=TRUE)
summary(boot.cnes, type="norm")
# cf., standard errors to those computed by summary(sem.cnes)

    ## End(Not run)
```

`mod.indices` *Modification Indices for Structural Equation Models*

Description

`mod.indices` calculates modification indices (score tests) and estimated parameter changes for the fixed and constrained parameters in a structural equation model. [Warning: This is a preliminary version: the computed values are a bit off the correct values.]

Usage

```
mod.indices(model, ...)

## S3 method for class 'sem':
mod.indices(model, ...)

## S3 method for class 'sem.modind':
print(x, n.largest=5, ...)

## S3 method for class 'sem.modind':
summary(object, round=2,
    print.matrices=c("both", "par.change", "mod.indices"), ...)
```

Arguments

`model`	an object of class `sem`, produced by the `sem` function.
`object, x`	an object of class `sem.modind`, produced by the `mod.indices` function.
`n.largest`	number of modification indices to print in each of the A and P matrices of the RAM model.
`round`	number of places to the right of the decimal point in printing modification indices.
`print.matrices`	which matrices to print: estimated changes in the fixed parameters, modification indices, or both (the default).
`...`	arguments to be passed down.

Details

Modification indices are one-df chi-square score statistics for the fixed and constrained parameters in a structural equation model. They may be regarded as an estimate of the improvement in the likelihood-ratio chi-square statistic for the model if the corresponding parameter is respecified as a free parameter. The `mod.indices` function also estimates the change in the value of a fixed or constrained parameter if the parameter is respecified as free. When several parameters are set equal, modification indices and estimated changes are given for all but the first. Modification indices and estimated parameter changes for currently free parameters are given as `NA`.

The method employed is described in Saris, Satorra, and Sorbom (1987) and Sorbom (1989).

Value

`mod.indices` returns an object of class `sem.modind` with the following elements:

`mod.A`	modification indices for the elements of the A matrix.
`mod.P`	modification indices for the elements of the P matrix.
`par.A`	estimated parameter changes for the elements of the A matrix.
`par.P`	estimated parameter changes for the elements of the P matrix.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

References

Sarris, W. E., Satorra, A., and Sorbom, D. (1987) The detection and correction of specification errors in structural equation models. Pp. 105–129 in Clogg, C. C. (ed.), *Sociological Methodology 1987*. American Sociological Association.

Sorbom, D. (1989) Model modification. *Psychometrika* **54**, 371–384.

See Also

`sem`

Examples

```
# This example is adapted from the SAS manual

S.wh <- read.moments(names=c('Anomia67','Powerless67','Anomia71',
                                    'Powerless71','Education','SEI'))
   11.834
    6.947    9.364
    6.819    5.091   12.532
    4.783    5.028    7.495    9.986
   -3.839   -3.889   -3.841   -3.625   9.610
  -21.899  -18.831  -21.748  -18.775  35.522  450.288

model.wh <- specify.model()
    Alienation67   ->  Anomia67,      NA,   1
    Alienation67   ->  Powerless67,   NA,   0.833
    Alienation71   ->  Anomia71,      NA,   1
    Alienation71   ->  Powerless71,   NA,   0.833
    SES            ->  Education,     NA,   1
    SES            ->  SEI,           lamb, NA
    SES            ->  Alienation67,  gam1, NA
    Alienation67   ->  Alienation71,  beta, NA
    SES            ->  Alienation71,  gam2, NA
    Anomia67       <-> Anomia67,      the1, NA
    Anomia71       <-> Anomia71,      the1, NA
    Powerless67    <-> Powerless67,   the2, NA
    Powerless71    <-> Powerless71,   the2, NA
    Education      <-> Education,     the3, NA
    SEI            <-> SEI,           the4, NA
    Anomia67       <-> Anomia71,      the5, NA
    Powerless67    <-> Powerless71,   the5, NA
    Alienation67   <-> Alienation67,  psi1, NA
    Alienation71   <-> Alienation71,  psi2, NA
    SES            <-> SES,           phi,  NA

sem.wh <- sem(model.wh, S.wh, 932)
mod.indices(sem.wh)

##   5 largest modification indices, A matrix:
##  Powerless67:Education    Anomia67:Education
##                 4.8736                3.8027
##        Powerless67:SES Education:Powerless67
##                 2.7608                2.4619
##         Anomia67:SES
##               2.3122
##
##    5 largest modification indices, P matrix:
##  Education:Powerless67    Education:Anomia67
##                 6.4028                4.5398
##        SES:Powerless67          SES:Anomia67
##                 2.7608                2.3122
##        SEI:Powerless67
##                 1.3185
```

path.diagram *Draw Path Diagram*

Description

path.diagram creates a description of the path diagram for a structural-equation-model object to be processed by the graph-drawing program *dot*; see Koutsofios and North (2002) and http://www.graphviz.org/.

Usage

```
path.diagram(model, ...)

## S3 method for class 'sem':
path.diagram(model, out.file, min.rank=NULL, max.rank=NULL,
    same.rank=NULL, variables=model$var.names, parameters=rownames(model$ram),
    ignore.double=TRUE, edge.labels=c("names", "values"), size=c(8, 8),
    node.font=c("Helvetica", 14), edge.font=c("Helvetica", 10),
    rank.direction=c("LR", "TB"), digits=2, ...)
```

Arguments

model	a structural-equation-model object produced by sem.
...	arguments to be passed down to path.diagram.sem.
out.file	a file to which to write the *dot* description of the path diagram; if not specified, the description is written to standard output (normally the R Console).
min.rank	a character string listing names of variables to be assigned minimum rank (order) in the graph; the names should be separated by commas.
max.rank	a character string listing names of variables to be assigned maximum rank in the graph; the names should be separated by commas.
same.rank	a character string or vector of character strings of variables to be assigned equivalent rank in the graph; names in each string should be separated by commas.
variables	variable names; defaults to the variable names in model. If specified, the variable names should be in the same order as in model.
parameters	parameter names; defaults to the parameter names in model. If specified, the parameter names should be in the same order as in model.
ignore.double	if TRUE, the default, double-headed arrows, representing variances and covariances, are not graphed.
edge.labels	"names" to label arrows with parameter names; "values" to label arrows with parameter estimates.
size	the size of the graph, in inches.
node.font	font name and point-size for printing variable names.

`edge.font` font name and point-size for printing arrow names or values.

`rank.direction` draw graph left-to-right, `"LR"`, the default, or top-to-bottom, `"TB"`.

`digits` number of digits after the decimal point (default, 2) to which to round parameter estimates.

Value

`NULL`: `path.diagram` is used for its side-effect, producing a graph description of the model.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

References

Koutsofios, E., and North, S. C. (2002) Drawing graphs with *dot*. http://www.graphviz.org/Documentation/dotguide.pdf.

See Also

sem

Examples

```
# The Duncan, Haller, and Portes Peer-Influences Model

R.DHP <- read.moments(diag=FALSE, names=c('ROccAsp', 'REdAsp', 'FOccAsp',
                'FEdAsp', 'RParAsp', 'RIQ', 'RSES', 'FSES', 'FIQ', 'FParAsp'))
    .6247
    .3269  .3669
    .4216  .3275  .6404
    .2137  .2742  .1124  .0839
    .4105  .4043  .2903  .2598  .1839
    .3240  .4047  .3054  .2786  .0489  .2220
    .2930  .2407  .4105  .3607  .0186  .1861  .2707
    .2995  .2863  .5191  .5007  .0782  .3355  .2302  .2950
    .0760  .0702  .2784  .1988  .1147  .1021  .0931 -.0438  .2087

model.dhp <- specify.model()
    RParAsp  -> RGenAsp, gam11,  NA
    RIQ      -> RGenAsp, gam12,  NA
    RSES     -> RGenAsp, gam13,  NA
    FSES     -> RGenAsp, gam14,  NA
    RSES     -> FGenAsp, gam23,  NA
    FSES     -> FGenAsp, gam24,  NA
    FIQ      -> FGenAsp, gam25,  NA
    FParAsp  -> FGenAsp, gam26,  NA
    FGenAsp  -> RGenAsp, beta12, NA
    RGenAsp  -> FGenAsp, beta21, NA
```

```
    RGenAsp  -> ROccAsp,  NA,       1
    RGenAsp  -> REdAsp,  lam21,  NA
    FGenAsp  -> FOccAsp,  NA,       1
    FGenAsp  -> FEdAsp,  lam42,  NA
    RGenAsp <-> RGenAsp, ps11,   NA
    FGenAsp <-> FGenAsp, ps22,   NA
    RGenAsp <-> FGenAsp, ps12,   NA
    ROccAsp <-> ROccAsp, theta1, NA
    REdAsp  <-> REdAsp,  theta2, NA
    FOccAsp <-> FOccAsp, theta3, NA
    FEdAsp  <-> FEdAsp,  theta4, NA

sem.dhp <- sem(model.dhp, R.DHP, 329,
    fixed.x=c('RParAsp', 'RIQ', 'RSES', 'FSES', 'FIQ', 'FParAsp'))

path.diagram(sem.dhp, min.rank='RIQ, RSES, RParAsp, FParAsp, FSES, FIQ',
    max.rank='ROccAsp, REdAsp, FEdAsp, FOccAsp')

##   digraph "sem.dhp" {
##   rankdir=LR;
##   size="8,8";
##   node [fontname="Helvetica" fontsize=14 shape=box];
##   edge [fontname="Helvetica" fontsize=10];
##   center=1;
##   {rank=min "RIQ" "RSES" "RParAsp" "FParAsp" "FSES" "FIQ"}
##   {rank=max "ROccAsp" "REdAsp" "FEdAsp" "FOccAsp"}
##   "RGenAsp" [shape=ellipse]
##   "FGenAsp" [shape=ellipse]
##   "RParAsp" -> "RGenAsp" [label="gam11"];
##   "RIQ" -> "RGenAsp" [label="gam12"];
##   "RSES" -> "RGenAsp" [label="gam13"];
##   "FSES" -> "RGenAsp" [label="gam14"];
##   "RSES" -> "FGenAsp" [label="gam23"];
##   "FSES" -> "FGenAsp" [label="gam24"];
##   "FIQ" -> "FGenAsp" [label="gam25"];
##   "FParAsp" -> "FGenAsp" [label="gam26"];
##   "FGenAsp" -> "RGenAsp" [label="beta12"];
##   "RGenAsp" -> "FGenAsp" [label="beta21"];
##   "RGenAsp" -> "ROccAsp" [label=""];
##   "RGenAsp" -> "REdAsp" [label="lam21"];
##   "FGenAsp" -> "FOccAsp" [label=""];
##   "FGenAsp" -> "FEdAsp" [label="lam42"];
##   }
```

`ram` *RAM Matrix for a Structural-Equation Model*

Description

Print the labelled RAM definition matrix for a structural-equation model fit by `sem`.

Usage

```
ram(object, digits=5, startvalues=FALSE)
```

Arguments

object	an object of class sem returned by the sem function.
digits	number of digits for printed output.
startvalues	if TRUE, start values for parameters are printed; otherwise, the parameter estimates are printed; the default is FALSE.

Value

A data frame containing the labelled RAM definition matrix, which is normally just printed.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

See Also

sem

Examples

```
# ------------- assumes that Duncan, Haller and Portes peer-influences model
# -------------     has been fit and is in sem.dhp

    ## Not run:
ram(sem.dhp)

##        heads to from parameter  estimate                arrow
##            1  1   11         0  1.000000 ROccAsp <--- RGenAsp
## lam21      1  2   11         1  1.062673  REdAsp <--- RGenAsp
##            1  3   12         0  1.000000 FOccAsp <--- FGenAsp
## lam42      1  4   12         2  0.929732  FEdAsp <--- FGenAsp
## gam11      1 11    5         3  0.161220 RGenAsp <--- RParAsp
## gam12      1 11    6         4  0.249647     RGenAsp <--- RIQ
## gam13      1 11    7         5  0.218402    RGenAsp <--- RSES
## gam14      1 11    8         6  0.071836    RGenAsp <--- FSES
## gam23      1 12    7         7  0.061879    FGenAsp <--- RSES
## gam24      1 12    8         8  0.228863    FGenAsp <--- FSES
## gam25      1 12    9         9  0.349030     FGenAsp <--- FIQ
## gam26      1 12   10        10  0.159529 FGenAsp <--- FParAsp
## beta12     1 11   12        11  0.184245 RGenAsp <--- FGenAsp
## beta21     1 12   11        12  0.235502 FGenAsp <--- RGenAsp
## theta1     2  1    1        13  0.412143 ROccAsp <--> ROccAsp
## theta2     2  2    2        14  0.336146   REdAsp <--> REdAsp
## theta3     2  3    3        15  0.311197 FOccAsp <--> FOccAsp
## theta4     2  4    4        16  0.404601   FEdAsp <--> FEdAsp
## psi11      2 11   11        17  0.280987 RGenAsp <--> RGenAsp
```

```
## psi22      2 12   12        18  0.263832 FGenAsp <--> FGenAsp
## psi12      2 11   12        19 -0.022620 RGenAsp <--> FGenAsp
##            2  5    5         0  1.000000 RParAsp <--> RParAsp
##            2  6    5         0  0.183900     RIQ <--> RParAsp
##            2  6    6         0  1.000000         RIQ <--> RIQ
##            2  7    5         0  0.048900    RSES <--> RParAsp
##            2  7    6         0  0.222000        RSES <--> RIQ
##            2  7    7         0  1.000000       RSES <--> RSES
##            2  8    5         0  0.018600    FSES <--> RParAsp
##            2  8    6         0  0.186100        FSES <--> RIQ
##            2  8    7         0  0.270700       FSES <--> RSES
##            2  8    8         0  1.000000       FSES <--> FSES
##            2  9    5         0  0.078200     FIQ <--> RParAsp
##            2  9    6         0  0.335500         FIQ <--> RIQ
##            2  9    7         0  0.230200        FIQ <--> RSES
##            2  9    8         0  0.295000        FIQ <--> FSES
##            2  9    9         0  1.000000         FIQ <--> FIQ
##            2 10    5         0  0.114700 FParAsp <--> RParAsp
##            2 10    6         0  0.102100     FParAsp <--> RIQ
##            2 10    7         0  0.093100    FParAsp <--> RSES
##            2 10    8         0 -0.043800    FParAsp <--> FSES
##            2 10    9         0  0.208700     FParAsp <--> FIQ
##            2 10   10         0  1.000000 FParAsp <--> FParAsp
    ## End(Not run)
```

`raw.moments` *Compute Raw Moments Matrix*

Description

Computes the "uncorrected" sum-of-squares-and-products matrix divided by the number of observations.

Usage

```
raw.moments(object, ...)

## S3 method for class 'formula':
raw.moments(formula, data, subset, na.action,
    contrasts = NULL, ...)

## Default S3 method:
raw.moments(object, ...)

cov2raw(cov, mean, N, sd)

## S3 method for class 'rawmoments':
print(x, ...)
```

Arguments

object	a one-sided model formula or an object coercible to a numeric matrix.
formula	a one-sided model formula specifying the model matrix for which raw moments are to be computed; note that a constant is included if it is not explicitly suppressed by putting -1 in the formula.
data	an optional data frame containing the variables in the formula. By default the variables are taken from the environment from which raw.moments is called.
subset	an optional vector specifying a subset of observations to be used in computing moments.
na.action	a function that indicates what should happen when the data contain NAs. The default is set by the na.action option.
contrasts	an optional list. See the contrasts.arg of model.matrix.default
cov	a covariance or correlation matrix.
mean	a vector of means.
N	the number of observations on which the covariances or correlations are based.
sd	an optional vector of standard deviations, to be given if cov is a correlation matrix.
x	an object of class rawmoments to print.
...	arguments passed down.

Value

raw.moments and cov2raw return an object of class rawmoments, which is simply a matrix with an attribute "N" that contains the number of observations on which the moments are based.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

See Also

sem

Examples

```
data(Kmenta)
raw.moments(cbind(1, Kmenta))
raw.moments(~ Q + P + D + F + A, data=Kmenta)

Cov <- with(Kmenta, cov(cbind(Q, P, D, F, A)))
cov2raw(Cov, colMeans(Kmenta), nrow(Kmenta))
```

`read.moments` *Input a Covariance, Correlation, or Raw Moment Matrix*

Description

This functions makes it simpler to input covariance, correlation, and raw-moment matrices to be analyzed by the `sem` function. The matrix is input in lower-triangular form on as many lines as is convenient, omitting the above-diagonal elements. The elements on the diagonal may also optionally be omitted, in which case they are taken to be 1.

Usage

```
read.moments(file = "", diag = TRUE,
    names = as.character(paste("X", 1:n, sep = "")))
```

Arguments

`file`	The (quoted) file from which to read the model specification, including the path to the file if it is not in the current directory. If `""` (the default), then the specification is read from the standard input stream, and is terminated by a blank line.
`diag`	If `TRUE` (the default), then the input matrix includes diagonal elements.
`names`	a character vector containing the names of the variables, to label the rows and columns of the moment matrix.

Value

Returns a lower-triangular matrix (i.e., with zeroes above the main diagonal) suitable for input to `sem`.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

See Also

`sem`

Examples

```
    ## Not run:
R.DHP <- read.moments(diag=FALSE, names=c("ROccAsp", "REdAsp", "FOccAsp",
                "FEdAsp", "RParAsp", "RIQ", "RSES", "FSES", "FIQ", "FParAsp"))
    .6247
    .3269  .3669
    .4216  .3275  .6404
    .2137  .2742  .1124  .0839
    .4105  .4043  .2903  .2598  .1839
```

```
     .3240  .4047  .3054  .2786  .0489  .2220
     .2930  .2407  .4105  .3607  .0186  .1861  .2707
     .2995  .2863  .5191  .5007  .0782  .3355  .2302  .2950
     .0760  .0702  .2784  .1988  .1147  .1021  .0931 -.0438  .2087

R.DHP

## End(Not run)
```

`residuals.sem` *Residual Covariances for a Structural Equation Model*

Description

These functions compute residual covariances, variance-standardized residual covariances, and normalized residual covariances for the observed variables in a structural-equation model fit by `sem`.

Usage

```
## S3 method for class 'sem':
residuals(object, ...)

standardized.residuals(object, ...)

## S3 method for class 'sem':
standardized.residuals(object, ...)

normalized.residuals(object, ...)

## S3 method for class 'sem':
normalized.residuals(object, ...)
```

Arguments

`object`	an object of class `sem` returned by the `sem` function.
`...`	not for the user.

Details

Residuals are defined as $S - C$, where S is the sample covariance matrix of the observed variables and C is the model-reproduced covariance matrix. The *standardized* residual covariance for a pair of variables divides the residual covariance by the product of the sample standard deviations of the two variables, $(s_{ij} - c_{ij})/(s_{ii}s_{jj})^{1/2}$. The *normalized* residual is given by

$$\frac{s_{ij} - c_{ij}}{[(c_{ii}c_{jj} - c_{ij}^2)/N^*]^{1/2}}$$

where N^* is the number of observations minus one if the model is fit to a covariance matrix, or the number of observations if it is fit to a raw moment matrix.

Value

Each function returns a matrix of residuals.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

References

Bollen, K. A. (1989) *Structural Equations With Latent Variables.* Wiley.

See Also

sem

Examples

```
    ## Not run:
# ------------- assumes that Duncan, Haller and Portes peer-influences model
# -------------     has been fit and is in sem.dhp.1

residuals(sem.dhp.1)

##                  ROccAsp        REdAsp       FOccAsp        FEdAsp
##    ROccAsp  4.156103e-07 -2.896368e-07 -2.866110e-02  9.102874e-02
##    REdAsp  -2.896368e-07 -1.541581e-06 -1.094841e-02 -2.379215e-02
##    FOccAsp -2.866110e-02 -1.094841e-02 -3.740356e-06 -9.564103e-07
##
##    . . .
##
##                      FIQ       FParAsp
##    ROccAsp  3.467853e-02 -3.187309e-02
##    REdAsp   4.878970e-03 -4.443480e-02
##    FOccAsp -7.354686e-03  2.488120e-02
##    FEdAsp   1.124604e-02 -3.690078e-02
##    RParAsp  2.775558e-17  5.551115e-17
##    RIQ      2.220446e-16  6.938894e-17
##    RSES     0.000000e+00 -1.387779e-17
##    FSES     1.110223e-16 -2.775558e-17
##    FIQ      4.440892e-16  1.110223e-16
##    FParAsp  1.110223e-16  4.440892e-16

normalized.residuals(sem.dhp.1)

##                  ROccAsp        REdAsp       FOccAsp        FEdAsp
##    ROccAsp  5.330519e-06 -4.455587e-06 -4.898232e-01  1.567678e+00
##    REdAsp  -4.455587e-06 -1.977191e-05 -1.857670e-01 -4.071582e-01
##    FOccAsp -4.898232e-01 -1.857670e-01 -4.797271e-05 -1.460881e-05
##
##    . . .
##
##                      FIQ       FParAsp
```

```
##    ROccAsp  6.080514e-01 -5.747909e-01
##    REdAsp   8.518738e-02 -8.007295e-01
##    FOccAsp -1.180429e-01  4.374639e-01
##    FEdAsp   1.832159e-01 -6.514685e-01
##    RParAsp  5.019082e-16  1.000322e-15
##    RIQ      3.818356e-15  1.252092e-15
##    RSES     0.000000e+00 -2.506364e-16
##    FSES     1.931472e-15 -5.029583e-16
##    FIQ      5.695780e-15  1.971289e-15
##    FParAsp  1.971289e-15  5.695780e-15

standardized.residuals(sem.dhp.1)

##                   ROccAsp        REdAsp       FOccAsp        FEdAsp
##    ROccAsp  4.156103e-07 -2.896368e-07 -2.866110e-02  9.102874e-02
##    REdAsp  -2.896368e-07 -1.541581e-06 -1.094841e-02 -2.379215e-02
##    FOccAsp -2.866110e-02 -1.094841e-02 -3.740356e-06 -9.564103e-07
##
##    . . .
##
##                      FIQ       FParAsp
##    ROccAsp  3.467853e-02 -3.187309e-02
##    REdAsp   4.878970e-03 -4.443480e-02
##    FOccAsp -7.354686e-03  2.488120e-02
##    FEdAsp   1.124604e-02 -3.690078e-02
##    RParAsp  2.775558e-17  5.551115e-17
##    RIQ      2.220446e-16  6.938894e-17
##    RSES     0.000000e+00 -1.387779e-17
##    FSES     1.110223e-16 -2.775558e-17
##    FIQ      4.440892e-16  1.110223e-16
##    FParAsp  1.110223e-16  4.440892e-16
    ## End(Not run)
```

`sem` *General Structural Equation Models*

Description

`sem` fits general structural equation models (with both observed and unobserved variables) by the method of maximum likelihood, assuming multinormal errors. Observed variables are also called *indicators* or *manifest variables*; unobserved variables are also called *factors* or *latent variables*. Normally, the generic function (`sem`) would be used.

Usage

```
sem(ram, ...)

## S3 method for class 'mod':
sem(ram, S, N, obs.variables=rownames(S), fixed.x=NULL, debug=FALSE, ...)
```

```
## Default S3 method:
sem(ram, S, N, param.names = paste("Param", 1:t, sep = ""),
    var.names = paste("V", 1:m, sep = ""), fixed.x = NULL, raw=FALSE,
    debug = FALSE, analytic.gradient = TRUE, warn = FALSE, maxiter = 500,
    par.size=c('ones', 'startvalues'), refit=TRUE, start.tol=1E-6, ...)

startvalues(S, ram, debug = FALSE, tol=1E-6)

## S3 method for class 'sem':
print(x, ...)

## S3 method for class 'sem':
summary(object, digits=5, conf.level=0.9, ...)

## S3 method for class 'sem':
deviance(object, ...)

## S3 method for class 'sem':
df.residual(object, ...)

## S3 method for class 'sem':
anova(object, model.2, ...)

## S3 method for class 'sem':
coef(object, ...)

## S3 method for class 'sem':
vcov(object, ...)
```

Arguments

ram	RAM specification, which is a simple encoding of the path diagram for the model. The ram matrix may be given either in symbolic form (as a mod object, as returned by the specify.model function, or as a character matrix), invoking sem.mod, which calls sem.default after setting up the model, or (less conveniently) in numeric form, invoking sem.default directly (see **Details** below).
S	covariance matrix among observed variables; may be input as a symmetric matrix, or as a lower- or upper-triangular matrix. S may also be a raw (i.e., "uncorrected") moment matrix — that is, a sum-of-squares-and-products matrix divided by N. This form of input is useful for fitting models with intercepts, in which case the moment matrix should include the mean square and cross-products for a unit variable all of whose entries are 1; of course, the raw mean square for the unit variable is 1. Raw-moment matrices may be computed by raw.moments. If the ram argument is given in symbolic form, then the observed-variable covariance or raw-moment matrix may contain variables that do not appear in the model, in which case a warning is printed.
N	number of observations on which the covariance matrix is based.

`obs.variables`
: names of observed variables, by default taken from the row names of the covariance matrix `S`.

`fixed.x`
: names (if the `ram` matrix is given in symbolic form) or indices (if it is in numeric form) of fixed exogenous variables. Specifying these obviates the necessity of having to fix the variances and covariances among these variables (and produces correct degrees of freedom for the model chisquare).

`raw`
: `TRUE` if `S` is a raw moment matrix, as opposed to a covariance matrix; the default is `FALSE`.

`debug`
: if `TRUE`, some information is printed to help you debug the symbolic model specification; for example, if a variable name is misspelled, `sem` will assume that the variable is a (new) latent variable. The default is `FALSE`.

`...`
: arguments to be passed down, including from `sem.default` to the `nlm` optimizer.

`param.names`
: names of the t free parameters, given in their numerical order; default names are `Param1`, ..., `Param`*t*. Note: Should not be specified when the `ram` matrix is given in symbolic form.

`var.names`
: names of the m entries of the v vector (typically the observed and latent variables — see below), given in their numerical order; default names are `Var1`, ..., `Var`*m*. Note: Should not be specified when the `ram` matrix is given in symbolic form.

`analytic.gradient`
: if `TRUE` (the default), then analytic first derivatives are used in the maximization of the likelihood; otherwise numeric derivatives are used.

`warn`
: if `TRUE`, warnings produced by the optimization function will be printed. This should generally not be necessary, since `sem` prints its own warning, and saves information about convergence. The default is `FALSE`.

`maxiter`
: the maximum number of iterations for the optimization performed by the `nlm` function, to be passed to it via its `iterlim` argument.

`par.size`
: the anticipated size of the free parameters; if `"ones"`, a vector of ones is used; if `"startvalues"`, taken from the start values. You can try changing this argument if you encounter convergence problems. The default is `"startvalues"` if the largest input variance is at least 100 times the smallest, and `"ones"` otherwise.

`refit`
: if `TRUE` (the default), attempt to refit the model eliminating apparently aliased parameters if under-identification is detected.

`start.tol, tol`
: if the magnitude of an automatic start value is less than `start.tol`, then it is set to `start.tol`; defaults to 1E-6.

`object, x`
: an object of class `sem` returned by the `sem` function.

`digits`
: number of digits for printed output.

`conf.level`
: level for confidence interval for the RMSEA index (default is .9).

`model.2`
: an object of class `sem` returned by the `sem` function, to be compared by a likelihood-ratio test to `object`; the two models must be fit to the same data.

Details

The model is set up using RAM ('reticular action model' – don't ask!) notation – a simple format for specifying general structural equation models by coding the 'arrows' in the path diagram for the model (see, e.g., McArdle and McDonald, 1984).

The variables in the v vector in the model (typically, the observed and unobserved variables, but not error variables) are numbered from 1 to m. the RAM matrix contains one row for each (free or constrained) parameter of the model, and may be specified either in symbolic format or in numeric format.

A symbolic `ram` matrix consists of three columns, as follows:

1. **Arrow specification:** This is a simple formula, of the form `"A -> B"` or, equivalently, `"B <- A"` for a regression coefficient (i.e., a single-headed or directional arrow); `"A <-> A"` for a variance or `"A <-> B"` for a covariance (i.e., a double-headed or bidirectional arrow). Here, `A` and `B` are variable names in the model. If a name does not correspond to an observed variable, then it is assumed to be a latent variable. Spaces can appear freely in an arrow specification, and there can be any number of hyphens in the arrows, including zero: Thus, e.g., `"A->B"`, `"A -> B"`, and `"A>B"` are all legitimate and equivalent.
2. **Parameter name:** The name of the regression coefficient, variance, or covariance specified by the arrow. Assigning the same name to two or more arrows results in an equality constraint. Specifying the parameter name as `NA` produces a fixed parameter.
3. **Value:** start value for a free parameter or value of a fixed parameter. If given as `NA`, `sem` will compute the start value.

It is simplest to construct the RAM matrix with the `specify.model` function, which returns an object of class `mod`. This process is illustrated in the examples below.

A numeric `ram` matrix consists of five columns, as follows:

1. **Number of arrow heads:** 1 (directed arrow) or 2 (covariance).
2. **Arrow *to*:** index of the variable at the head of a directional arrow, or at one end of a bidirectional arrow. Observed variables should be assigned the numbers 1 to n, where n is the number of rows/columns in the covariance matrix `S`, with the indices corresponding to the variables' positions in `S`. Variable indices above n represent latent variables.
3. **Arrow *from*:** the index of the variable at the tail of a directional arrow, or at the other end of a bidirectional arrow.
4. **Parameter number:** free parameters are numbered from 1 to t, but do not necessarily appear in consecutive order. Fixed parameters are given the number 0. Equality contraints are specified by assigning two or more parameters the same number.
5. **Value:** start value for a free parameter, or value of a fixed parameter. If given as `NA`, the program will compute a start value, by a slight modification of the method described by McDonald and Hartmann (1992). *Note:* In some circumstances, some start values are selected randomly; this might produce small differences in the parameter estimates when the program is rerun.

`sem` fits the model by calling the `nlm` optimizer to minimize the negative log-likelihood for the model. If `nlm` fails to converge, a warning message is printed.

The RAM formulation of the general structural equation model is given by the basic equation

$$v = Av + u$$

where v and u are vectors of random variables (observed or unobserved), and the parameter matrix A contains regression coefficients, symbolized by single-headed arrows in a path diagram. Another parameter matrix,

$$P = E(uu')$$

contains covariances among the elements of u (assuming that the elements of u have zero means). Usually v contains endogenous and exogenous observed and unobserved variables, but not error variables (see the examples below).

The `startvalues` function may be called directly, but is usually called by `sem.default`.

The `sem` methods for the generic functions `deviance` and `df.residual` functions return, respectively, minus twice the difference in the log-likelihoods for the fitted model and a saturated model, and the degrees of freedom associated with the deviance.

Value

`sem` returns an object of class `sem`, with the following elements:

`ram`	RAM matrix, including any rows generated for covariances among fixed exogenous variables; column 5 includes computed start values.
`coeff`	estimates of free parameters.
`criterion`	fitting criterion — minus twice the difference in the log-liklihood between the fitted model and a saturated model, divided by $N - 1$ (or N if the model is fit to a raw moment matrix).
`cov`	estimated asymptotic covariance matrix of parameter estimates.
`S`	observed covariance matrix.
`J`	RAM selection matrix, J, which picks out observed variables.
`C`	model-reproduced covariance matrix.
`A`	RAM A matrix.
`P`	RAM P matrix.
`n.fix`	number of fixed exogenous variables.
`n`	number of observed variables.
`N`	number of observations.
`m`	number of variables (observed plus unobserved).
`t`	number of free parameters.
`par.posn`	indices of free parameters.
`var.names`	vector of variable names.
`observed`	indices of observed variables.
`convergence`	convergence code returned by `nlm` (a code > 2 indicates a problem).
`iterations`	number of iterations performed.
`raw`	`TRUE` if the model is fit to a raw moment matrix, `FALSE` otherwise.
`chisqNull`	Unless the model is fit to a raw moment matrix, the chisquare value associated with a null model in which all of the observed variables are uncorrelated.

Warning

A common error is to fail to specify variance or covariance terms in the model, which are denoted by double-headed arrows, <->.

In general, every observed or latent variable in the model should be associated with a variance or error variance. This may be a free parameter to estimate or a fixed constant (as in the case of a latent exogenous variable for which you wish to fix the variance, e.g., to 1). Again in general, there will be an *error variance* associated with each endogenous variable in the model (i.e., each variable to which at least one single-headed arrow points — including observed indicators of latent variables), and a *variance* associated with each exogenous variable (i.e., each variable that appears only at the tail of single-headed arrows, never at the head).

To my knowledge, the only *apparent* exception to this rule is for observed variables that are declared to be fixed exogenous variables. In this case, the program generates the necessary (fixed-constant) variances and covariances automatically.

If there are missing variances, a warning message will be printed, and estimation will almost surely fail in some manner. Missing variances might well indicate that there are missing covariances too, but it is not possible to deduce this in a mechanical manner.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

References

Bollen, K. A. (1989) *Structural Equations With Latent Variables.* Wiley.

Bollen, K. A. and Long, J. S. (eds.) *Testing Structural Equation Models*, Sage.

McArdle, J. J. and Epstein, D. (1987) Latent growth curves within developmental structural equation models. *Child Development* **58**, 110–133.

McArdle, J. J. and McDonald, R. P. (1984) Some algebraic properties of the reticular action model. *British Journal of Mathematical and Statistical Psychology* **37**, 234–251.

McDonald, R. P. and Hartmann, W. M. (1992) A procedure for obtaining initial values of parameters in the RAM model. *Multivariate Behavioral Research* **27**, 57–76.

Raftery, A. E. (1993) Bayesian model selection in structural equation models. In Bollen, K. A. and Long, J. S. (eds.) *Testing Structural Equation Models*, Sage.

Raftery, A. E. (1995) Bayesian model selection in social research (with discussion). *Sociological Methodology* **25**, 111–196,

See Also

raw.moments, nlm

Examples

```
# Note: These examples can't be run via example() because the default file
#  argument of specify.model() and read.moments() requires that the model
#  specification and covariances, correlations, or raw moments be entered
```

```
#  at the command prompt. The examples can be copied and run in the R console,
#  however. See ?specify.model and ?read.moments for further information.

    ## Not run:

# ------------- Duncan, Haller and Portes peer-influences model ----------------------
# A nonrecursive SEM with unobserved endogenous variables and fixed exogenous variables

R.DHP <- read.moments(diag=FALSE, names=c("ROccAsp", "REdAsp", "FOccAsp",
                "FEdAsp", "RParAsp", "RIQ", "RSES", "FSES", "FIQ", "FParAsp"))
    .6247
    .3269  .3669
    .4216  .3275  .6404
    .2137  .2742  .1124  .0839
    .4105  .4043  .2903  .2598  .1839
    .3240  .4047  .3054  .2786  .0489  .2220
    .2930  .2407  .4105  .3607  .0186  .1861  .2707
    .2995  .2863  .5191  .5007  .0782  .3355  .2302  .2950
    .0760  .0702  .2784  .1988  .1147  .1021  .0931 -.0438  .2087

# Fit the model using a symbolic ram specification

model.dhp <- specify.model()
    RParAsp  -> RGenAsp, gam11,  NA
    RIQ      -> RGenAsp, gam12,  NA
    RSES     -> RGenAsp, gam13,  NA
    FSES     -> RGenAsp, gam14,  NA
    RSES     -> FGenAsp, gam23,  NA
    FSES     -> FGenAsp, gam24,  NA
    FIQ      -> FGenAsp, gam25,  NA
    FParAsp  -> FGenAsp, gam26,  NA
    FGenAsp  -> RGenAsp, beta12, NA
    RGenAsp  -> FGenAsp, beta21, NA
    RGenAsp  -> ROccAsp,  NA,     1
    RGenAsp  -> REdAsp,  lam21,  NA
    FGenAsp  -> FOccAsp,  NA,     1
    FGenAsp  -> FEdAsp,  lam42,  NA
    RGenAsp <-> RGenAsp, ps11,   NA
    FGenAsp <-> FGenAsp, ps22,   NA
    RGenAsp <-> FGenAsp, ps12,   NA
    ROccAsp <-> ROccAsp, theta1, NA
    REdAsp  <-> REdAsp,  theta2, NA
    FOccAsp <-> FOccAsp, theta3, NA
    FEdAsp  <-> FEdAsp,  theta4, NA

sem.dhp.1 <- sem(model.dhp, R.DHP, 329,
    fixed.x=c('RParAsp', 'RIQ', 'RSES', 'FSES', 'FIQ', 'FParAsp'))

summary(sem.dhp.1)

##     Model Chisquare =  26.697   Df =  15 Pr(>Chisq) = 0.031302
##     Chisquare (null model) =  872   Df =  45
##     Goodness-of-fit index =  0.98439
```

```
##     Adjusted goodness-of-fit index =  0.94275
##     RMSEA index =  0.048759   90 % CI: (0.014517, 0.07831)
##     Bentler-Bonnett NFI =  0.96938
##     Tucker-Lewis NNFI =  0.95757
##     Bentler CFI =  0.98586
##     SRMR =  0.020204
##     BIC =  -60.244
##
##     Normalized Residuals
##       Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    -0.7990 -0.1180  0.0000 -0.0120  0.0397  1.5700
##
##     Parameter Estimates
##           Estimate  Std Error z value Pr(>|z|)
##    gam11   0.161224 0.038487   4.1890 2.8019e-05 RGenAsp <--- RParAsp
##    gam12   0.249653 0.044580   5.6001 2.1428e-08 RGenAsp <--- RIQ
##    gam13   0.218404 0.043476   5.0235 5.0730e-07 RGenAsp <--- RSES
##    gam14   0.071843 0.050335   1.4273 1.5350e-01 RGenAsp <--- FSES
##    gam23   0.061894 0.051738   1.1963 2.3158e-01 FGenAsp <--- RSES
##    gam24   0.228868 0.044495   5.1437 2.6938e-07 FGenAsp <--- FSES
##    gam25   0.349039 0.044551   7.8346 4.6629e-15 FGenAsp <--- FIQ
##    gam26   0.159535 0.040129   3.9755 7.0224e-05 FGenAsp <--- FParAsp
##    beta12  0.184226 0.096207   1.9149 5.5506e-02 RGenAsp <--- FGenAsp
##    beta21  0.235458 0.119742   1.9664 4.9255e-02 FGenAsp <--- RGenAsp
##    lam21   1.062674 0.091967  11.5549 0.0000e+00 REdAsp <--- RGenAsp
##    lam42   0.929727 0.071152  13.0668 0.0000e+00 FEdAsp <--- FGenAsp
##    ps11    0.280987 0.046311   6.0674 1.2999e-09 RGenAsp <--> RGenAsp
##    ps22    0.263836 0.044902   5.8759 4.2067e-09 FGenAsp <--> FGenAsp
##    ps12   -0.022601 0.051649  -0.4376 6.6168e-01 FGenAsp <--> RGenAsp
##    theta1  0.412145 0.052211   7.8939 2.8866e-15 ROccAsp <--> ROccAsp
##    theta2  0.336148 0.053323   6.3040 2.9003e-10 REdAsp <--> REdAsp
##    theta3  0.311194 0.046665   6.6687 2.5800e-11 FOccAsp <--> FOccAsp
##    theta4  0.404604 0.046733   8.6578 0.0000e+00 FEdAsp <--> FEdAsp
##
##     Iterations =  28

# Fit the model using a numerical ram specification

ram.dhp <- matrix(c(
#               heads   to      from    param  start
                1,       1,     11,      0,     1,
                1,       2,     11,      1,     NA, # lam21
                1,       3,     12,      0,     1,
                1,       4,     12,      2,     NA, # lam42
                1,      11,      5,      3,     NA, # gam11
                1,      11,      6,      4,     NA, # gam12
                1,      11,      7,      5,     NA, # gam13
                1,      11,      8,      6,     NA, # gam14
                1,      12,      7,      7,     NA, # gam23
                1,      12,      8,      8,     NA, # gam24
                1,      12,      9,      9,     NA, # gam25
                1,      12,     10,     10,     NA, # gam26
                1,      11,     12,     11,     NA, # beta12
```

```
                1,      12,     11,     12,     NA, # beta21
                2,       1,      1,     13,     NA, # theta1
                2,       2,      2,     14,     NA, # theta2
                2,       3,      3,     15,     NA, # theta3
                2,       4,      4,     16,     NA, # theta4
                2,      11,     11,     17,     NA, # psi11
                2,      12,     12,     18,     NA, # psi22
                2,      11,     12,     19,     NA  # psi12
                ), ncol=5, byrow=TRUE)

params.dhp <- c('lam21', 'lam42', 'gam11', 'gam12', 'gam13', 'gam14',
                'gam23',  'gam24',  'gam25',  'gam26',
                'beta12', 'beta21', 'theta1', 'theta2', 'theta3', 'theta4',
                'psi11', 'psi22', 'psi12')

vars.dhp <- c('ROccAsp', 'REdAsp', 'FOccAsp', 'FEdAsp', 'RParAsp', 'RIQ',
              'RSES', 'FSES', 'FIQ', 'FParAsp', 'RGenAsp', 'FGenAsp')

sem.dhp.2 <- sem(ram.dhp, R.DHP, 329, params.dhp, vars.dhp, fixed.x=5:10)

summary(sem.dhp.2)

##     Model Chisquare =  26.697   Df =  15 Pr(>Chisq) = 0.031302
##     Chisquare (null model) =  872   Df =  45
##     Goodness-of-fit index =  0.98439
##     Adjusted goodness-of-fit index =  0.94275
##     RMSEA index =  0.048759   90 % CI: (0.014517, 0.07831)
##     Bentler-Bonnett NFI =  0.96938
##     Tucker-Lewis NNFI =  0.95757
##     Bentler CFI =  0.98586
##     SRMR =  0.020204
##     BIC =  -60.244
##
##     Normalized Residuals
##       Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    -0.7990 -0.1180  0.0000 -0.0120  0.0397  1.5700
##
##     Parameter Estimates
##           Estimate  Std Error z value Pr(>|z|)
##    lam21   1.062674 0.091967  11.5549 0.0000e+00 REdAsp <--- RGenAsp
##    lam42   0.929727 0.071152  13.0668 0.0000e+00 FEdAsp <--- FGenAsp
##    gam11   0.161224 0.038487   4.1890 2.8019e-05 RGenAsp <--- RParAsp
##    gam12   0.249653 0.044580   5.6001 2.1428e-08 RGenAsp <--- RIQ
##    gam13   0.218404 0.043476   5.0235 5.0730e-07 RGenAsp <--- RSES
##    gam14   0.071843 0.050335   1.4273 1.5350e-01 RGenAsp <--- FSES
##    gam23   0.061894 0.051738   1.1963 2.3158e-01 FGenAsp <--- RSES
##    gam24   0.228868 0.044495   5.1437 2.6938e-07 FGenAsp <--- FSES
##    gam25   0.349039 0.044551   7.8346 4.6629e-15 FGenAsp <--- FIQ
##    gam26   0.159535 0.040129   3.9755 7.0224e-05 FGenAsp <--- FParAsp
##    beta12  0.184226 0.096207   1.9149 5.5506e-02 RGenAsp <--- FGenAsp
##    beta21  0.235458 0.119742   1.9664 4.9255e-02 FGenAsp <--- RGenAsp
##    theta1  0.412145 0.052211   7.8939 2.8866e-15 ROccAsp <--> ROccAsp
##    theta2  0.336148 0.053323   6.3040 2.9003e-10 REdAsp <--> REdAsp
```

```
##    theta3  0.311194 0.046665   6.6687 2.5800e-11 FOccAsp <--> FOccAsp
##    theta4  0.404604 0.046733   8.6578 0.0000e+00 FEdAsp <--> FEdAsp
##    psi11   0.280987 0.046311   6.0674 1.2999e-09 RGenAsp <--> RGenAsp
##    psi22   0.263836 0.044902   5.8759 4.2067e-09 FGenAsp <--> FGenAsp
##    psi12  -0.022601 0.051649  -0.4376 6.6168e-01 RGenAsp <--> FGenAsp

##   Iterations =  28

# -------------------- Wheaton et al. alienation data ----------------------


S.wh <- read.moments(names=c('Anomia67','Powerless67','Anomia71',
                                  'Powerless71','Education','SEI'))
   11.834
    6.947    9.364
    6.819    5.091   12.532
    4.783    5.028    7.495    9.986
   -3.839   -3.889   -3.841   -3.625   9.610
  -21.899  -18.831  -21.748  -18.775  35.522  450.288

# This is the model in the SAS manual for PROC CALIS: A Recursive SEM with
# latent endogenous and exogenous variables.
# Curiously, both factor loadings for two of the latent variables are fixed.

model.wh.1 <- specify.model()
    Alienation67   ->  Anomia67,      NA,     1
    Alienation67   ->  Powerless67,   NA,     0.833
    Alienation71   ->  Anomia71,      NA,     1
    Alienation71   ->  Powerless71,   NA,     0.833
    SES            ->  Education,     NA,     1
    SES            ->  SEI,           lamb,   NA
    SES            ->  Alienation67,  gam1,   NA
    Alienation67   ->  Alienation71,  beta,   NA
    SES            ->  Alienation71,  gam2,   NA
    Anomia67       <-> Anomia67,      the1,   NA
    Anomia71       <-> Anomia71,      the1,   NA
    Powerless67    <-> Powerless67,   the2,   NA
    Powerless71    <-> Powerless71,   the2,   NA
    Education      <-> Education,     the3,   NA
    SEI            <-> SEI,           the4,   NA
    Anomia67       <-> Anomia71,      the5,   NA
    Powerless67    <-> Powerless71,   the5,   NA
    Alienation67   <-> Alienation67,  psi1,   NA
    Alienation71   <-> Alienation71,  psi2,   NA
    SES            <-> SES,           phi,    NA


sem.wh.1 <- sem(model.wh.1, S.wh, 932)

summary(sem.wh.1)

##     Model Chisquare =  13.485   Df =  9 Pr(>Chisq) = 0.14186
##     Chisquare (null model) =  2131.4   Df =  15
```

```
##     Goodness-of-fit index =  0.99527
##     Adjusted goodness-of-fit index =  0.98896
##     RMSEA index =  0.023136   90 % CI: (NA, 0.046997)
##     Bentler-Bonnett NFI =  0.99367
##     Tucker-Lewis NNFI =  0.99647
##     Bentler CFI =  0.99788
##     SRMR =  0.014998
##     BIC =  -48.051
##
##     Normalized Residuals
##        Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    -1.26000 -0.13100  0.00014 -0.02870  0.11400  0.87400
##
##     Parameter Estimates
##         Estimate  Std Error z value  Pr(>|z|)
##    lamb   5.36880  0.433982  12.3710 0.0000e+00 SEI <--- SES
##    gam1  -0.62994  0.056128 -11.2233 0.0000e+00 Alienation67 <--- SES
##    beta   0.59312  0.046820  12.6680 0.0000e+00 Alienation71 <--- Alienation67
##    gam2  -0.24086  0.055202  -4.3632 1.2817e-05 Alienation71 <--- SES
##    the1   3.60787  0.200589  17.9864 0.0000e+00 Anomia67 <--> Anomia67
##    the2   3.59494  0.165234  21.7567 0.0000e+00 Powerless67 <--> Powerless67
##    the3   2.99366  0.498972   5.9996 1.9774e-09 Education <--> Education
##    the4 259.57583 18.321121  14.1681 0.0000e+00 SEI <--> SEI
##    the5   0.90579  0.121710   7.4422 9.9032e-14 Anomia71 <--> Anomia67
##    psi1   5.67050  0.422906  13.4084 0.0000e+00 Alienation67 <--> Alienation67
##    psi2   4.51481  0.334993  13.4773 0.0000e+00 Alienation71 <--> Alienation71
##    phi    6.61632  0.639506  10.3460 0.0000e+00 SES <--> SES
##
##     Iterations =  78

# The same model, but treating one loading for each latent variable as free
# (and equal to each other).

model.wh.2 <- specify.model()
    Alienation67   ->  Anomia67,      NA,        1
    Alienation67   ->  Powerless67,   lamby,    NA
    Alienation71   ->  Anomia71,      NA,        1
    Alienation71   ->  Powerless71,   lamby,    NA
    SES            ->  Education,     NA,        1
    SES            ->  SEI,           lambx,    NA
    SES            ->  Alienation67,  gam1,     NA
    Alienation67   ->  Alienation71,  beta,     NA
    SES            ->  Alienation71,  gam2,     NA
    Anomia67       <-> Anomia67,      the1,     NA
    Anomia71       <-> Anomia71,      the1,     NA
    Powerless67    <-> Powerless67,   the2,     NA
    Powerless71    <-> Powerless71,   the2,     NA
    Education      <-> Education,     the3,     NA
    SEI            <-> SEI,           the4,     NA
    Anomia67       <-> Anomia71,      the5,     NA
    Powerless67    <-> Powerless71,   the5,     NA
    Alienation67   <-> Alienation67,  psi1,     NA
    Alienation71   <-> Alienation71,  psi2,     NA
```

```
    SES            <-> SES,           phi,      NA

sem.wh.2 <- sem(model.wh.2, S.wh, 932)

summary(sem.wh.2)

##     Model Chisquare =  12.673   Df =  8 Pr(>Chisq) = 0.12360
##     Chisquare (null model) =  2131.4   Df =  15
##     Goodness-of-fit index =  0.99553
##     Adjusted goodness-of-fit index =  0.98828
##     RMSEA index =  0.025049   90 % CI: (NA, 0.04985)
##     Bentler-Bonnett NFI =  0.99405
##     Tucker-Lewis NNFI =  0.99586
##     Bentler CFI =  0.9978
##     SRMR =  0.012981
##     BIC =  -42.026
##
##     Normalized Residuals
##         Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##    -0.997000 -0.140000  0.000295 -0.028800  0.100000  0.759000
##
##     Parameter Estimates
##          Estimate  Std Error z value  Pr(>|z|)
##    lamby   0.86261  0.033383  25.8402 0.0000e+00 Powerless67 <--- Alienation67
##    lambx   5.35302  0.432591  12.3743 0.0000e+00 SEI <--- SES
##    gam1   -0.62129  0.056142 -11.0663 0.0000e+00 Alienation67 <--- SES
##    beta    0.59428  0.047040  12.6335 0.0000e+00 Alienation71 <--- Alienation67
##    gam2   -0.23580  0.054684  -4.3121 1.6173e-05 Alienation71 <--- SES
##    the1    3.74499  0.249823  14.9906 0.0000e+00 Anomia67 <--> Anomia67
##    the2    3.49378  0.200754  17.4033 0.0000e+00 Powerless67 <--> Powerless67
##    the3    2.97409  0.499661   5.9522 2.6456e-09 Education <--> Education
##    the4  260.13252 18.298141  14.2163 0.0000e+00 SEI <--> SEI
##    the5    0.90377  0.121818   7.4190 1.1791e-13 Anomia71 <--> Anomia67
##    psi1    5.47380  0.464073  11.7951 0.0000e+00 Alienation67 <--> Alienation67
##    psi2    4.36410  0.362722  12.0315 0.0000e+00 Alienation71 <--> Alienation71
##    phi     6.63576  0.640425  10.3615 0.0000e+00 SES <--> SES
##
##     Iterations =  79
##

# Compare the two models by a likelihood-ratio test:

anova(sem.wh.1, sem.wh.2)

##    LR Test for Difference Between Models
##
##            Model Df Model Chisq Df LR Chisq Pr(>Chisq)
##    Model 1        9     13.4851
##    Model 2        8     12.6731  1   0.8119     0.3676

# ---------------------- Thurstone data ---------------------------------------
#  Second-order confirmatory factor analysis, from the SAS manual for PROC CALIS
```

```
R.thur <- read.moments(diag=FALSE, names=c('Sentences','Vocabulary',
        'Sent.Completion','First.Letters','4.Letter.Words','Suffixes',
        'Letter.Series','Pedigrees', 'Letter.Group'))
    .828
    .776   .779
    .439   .493    .46
    .432   .464    .425   .674
    .447   .489    .443   .59    .541
    .447   .432    .401   .381    .402   .288
    .541   .537    .534   .35    .367   .32   .555
    .38   .358    .359   .424    .446   .325   .598   .452

model.thur <- specify.model()
    F1 -> Sentences,                       lam11, NA
    F1 -> Vocabulary,                      lam21, NA
    F1 -> Sent.Completion,                 lam31, NA
    F2 -> First.Letters,                   lam41, NA
    F2 -> 4.Letter.Words,                  lam52, NA
    F2 -> Suffixes,                        lam62, NA
    F3 -> Letter.Series,                   lam73, NA
    F3 -> Pedigrees,                       lam83, NA
    F3 -> Letter.Group,                    lam93, NA
    F4 -> F1,                              gam1,  NA
    F4 -> F2,                              gam2,  NA
    F4 -> F3,                              gam3,  NA
    Sentences <-> Sentences,               th1,   NA
    Vocabulary <-> Vocabulary,             th2,   NA
    Sent.Completion <-> Sent.Completion,   th3,   NA
    First.Letters <-> First.Letters,       th4,   NA
    4.Letter.Words <-> 4.Letter.Words,     th5,   NA
    Suffixes <-> Suffixes,                 th6,   NA
    Letter.Series <-> Letter.Series,       th7,   NA
    Pedigrees <-> Pedigrees,               th8,   NA
    Letter.Group <-> Letter.Group,         th9,   NA
    F1 <-> F1,                             NA,     1
    F2 <-> F2,                             NA,     1
    F3 <-> F3,                             NA,     1
    F4 <-> F4,                             NA,     1

sem.thur <- sem(model.thur, R.thur, 213)

summary(sem.thur)

##     Model Chisquare =  38.196   Df =  24 Pr(>Chisq) = 0.033101
##     Chisquare (null model) =  1101.9   Df =  36
##     Goodness-of-fit index =  0.95957
##     Adjusted goodness-of-fit index =  0.9242
##     RMSEA index =  0.052822   90 % CI: (0.015262, 0.083067)
##     Bentler-Bonnett NFI =  0.96534
##     Tucker-Lewis NNFI =  0.98002
##     Bentler CFI =  0.98668
##     SRMR =  0.043595
##     BIC =  -90.475
```

```
##
##     Normalized Residuals
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##    -9.72e-01 -4.16e-01 -4.20e-07  4.01e-02  9.39e-02  1.63e+00
##
##     Parameter Estimates
##          Estimate Std Error z value Pr(>|z|)
##    lam11 0.51512  0.064964  7.9293  2.2204e-15 Sentences <--- F1
##    lam21 0.52031  0.065162  7.9849  1.3323e-15 Vocabulary <--- F1
##    lam31 0.48743  0.062422  7.8087  5.7732e-15 Sent.Completion <--- F1
##    lam41 0.52112  0.063137  8.2538  2.2204e-16 First.Letters <--- F2
##    lam52 0.49707  0.059673  8.3298  0.0000e+00 4.Letter.Words <--- F2
##    lam62 0.43806  0.056479  7.7562  8.6597e-15 Suffixes <--- F2
##    lam73 0.45244  0.071371  6.3392  2.3100e-10 Letter.Series <--- F3
##    lam83 0.41729  0.061037  6.8367  8.1020e-12 Pedigrees <--- F3
##    lam93 0.40763  0.064524  6.3175  2.6584e-10 Letter.Group <--- F3
##    gam1  1.44381  0.264173  5.4654  4.6184e-08 F1 <--- F4
##    gam2  1.25383  0.216597  5.7888  7.0907e-09 F2 <--- F4
##    gam3  1.40655  0.279331  5.0354  4.7681e-07 F3 <--- F4
##    th1   0.18150  0.028400  6.3907  1.6517e-10 Sentences <--> Sentences
##    th2   0.16493  0.027797  5.9334  2.9678e-09 Vocabulary <--> Vocabulary
##    th3   0.26713  0.033468  7.9816  1.5543e-15 Sent.Completion <--> Sent.Completion
##    th4   0.30150  0.050686  5.9484  2.7073e-09 First.Letters <--> First.Letters
##    th5   0.36450  0.052358  6.9617  3.3618e-12 4.Letter.Words <--> 4.Letter.Words
##    th6   0.50642  0.059963  8.4455  0.0000e+00 Suffixes <--> Suffixes
##    th7   0.39033  0.061599  6.3367  2.3474e-10 Letter.Series <--> Letter.Series
##    th8   0.48137  0.065388  7.3618  1.8141e-13 Pedigrees <--> Pedigrees
##    th9   0.50510  0.065227  7.7437  9.5479e-15 Letter.Group <--> Letter.Group
##
##    Iterations =  54
##

#------------------------- Kerchoff/Kenney path analysis ----------------------
# An observed-variable recursive SEM from the LISREL manual

R.kerch <- read.moments(diag=FALSE, names=c('Intelligence','Siblings',
                        'FatherEd','FatherOcc','Grades','EducExp','OccupAsp'))
    -.100
     .277  -.152
     .250  -.108  .611
     .572  -.105  .294   .248
     .489  -.213  .446   .410   .597
     .335  -.153  .303   .331   .478   .651

model.kerch <- specify.model()
    Intelligence -> Grades,       gam51,    NA
    Siblings -> Grades,           gam52,    NA
    FatherEd -> Grades,           gam53,    NA
    FatherOcc -> Grades,          gam54,    NA
    Intelligence -> EducExp,      gam61,    NA
    Siblings -> EducExp,          gam62,    NA
    FatherEd -> EducExp,          gam63,    NA
    FatherOcc -> EducExp,         gam64,    NA
```

```
    Grades -> EducExp,            beta65,   NA
    Intelligence -> OccupAsp,     gam71,    NA
    Siblings -> OccupAsp,         gam72,    NA
    FatherEd -> OccupAsp,         gam73,    NA
    FatherOcc -> OccupAsp,        gam74,    NA
    Grades -> OccupAsp,           beta75,   NA
    EducExp -> OccupAsp,          beta76,   NA
    Grades <-> Grades,            psi5,     NA
    EducExp <-> EducExp,          psi6,     NA
    OccupAsp <-> OccupAsp,        psi7,     NA


sem.kerch <- sem(model.kerch, R.kerch, 737, fixed.x=c('Intelligence','Siblings',
    'FatherEd','FatherOcc'))

summary(sem.kerch)

##     Model Chisquare =  3.2685e-13   Df =  0 Pr(>Chisq) = NA
##     Chisquare (null model) =  1664.3   Df =  21
##     Goodness-of-fit index =  1
##     BIC =  3.2685e-13
##
##     Normalized Residuals
##         Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##    -4.28e-15  0.00e+00  0.00e+00  7.62e-16  1.47e-15  5.17e-15
##
##     Parameter Estimates
##           Estimate  Std Error z value  Pr(>|z|)
##    gam51   0.525902 0.031182  16.86530 0.0000e+00 Grades <--- Intelligence
##    gam52  -0.029942 0.030149  -0.99314 3.2064e-01 Grades <--- Siblings
##    gam53   0.118966 0.038259   3.10951 1.8740e-03 Grades <--- FatherEd
##    gam54   0.040603 0.037785   1.07456 2.8257e-01 Grades <--- FatherOcc
##    gam61   0.160270 0.032710   4.89979 9.5940e-07 EducExp <--- Intelligence
##    gam62  -0.111779 0.026876  -4.15899 3.1966e-05 EducExp <--- Siblings
##    gam63   0.172719 0.034306   5.03461 4.7882e-07 EducExp <--- FatherEd
##    gam64   0.151852 0.033688   4.50758 6.5571e-06 EducExp <--- FatherOcc
##    beta65  0.405150 0.032838  12.33799 0.0000e+00 EducExp <--- Grades
##    gam71  -0.039405 0.034500  -1.14215 2.5339e-01 OccupAsp <--- Intelligence
##    gam72  -0.018825 0.028222  -0.66700 5.0477e-01 OccupAsp <--- Siblings
##    gam73  -0.041333 0.036216  -1.14126 2.5376e-01 OccupAsp <--- FatherEd
##    gam74   0.099577 0.035446   2.80924 4.9658e-03 OccupAsp <--- FatherOcc
##    beta75  0.157912 0.037443   4.21738 2.4716e-05 OccupAsp <--- Grades
##    beta76  0.549593 0.038260  14.36486 0.0000e+00 OccupAsp <--- EducExp
##    psi5    0.650995 0.033946  19.17743 0.0000e+00 Grades <--> Grades
##    psi6    0.516652 0.026943  19.17590 0.0000e+00 EducExp <--> EducExp
##    psi7    0.556617 0.029026  19.17644 0.0000e+00 OccupAsp <--> OccupAsp
##
##     Iterations =  0

#------------------- McArdle/Epstein latent-growth-curve model -----------------
# This model, from McArdle and Epstein (1987, p.118), illustrates the use of a
# raw moment matrix to fit a model with an intercept. (The example was suggested
# by Mike Stoolmiller.)
```

```
M.McArdle <- read.moments(names=c('WISC1', 'WISC2', 'WISC3', 'WISC4', 'UNIT'))
    365.661
    503.175     719.905
    675.656     958.479    1303.392
    890.680    1265.846    1712.475    2278.257
     18.034      25.819      35.255      46.593     1.000

mod.McArdle <- specify.model()
    C -> WISC1, NA, 6.07
    C -> WISC2, B2, NA
    C -> WISC3, B3, NA
    C -> WISC4, B4, NA
    UNIT -> C, Mc, NA
    C <-> C, Vc, NA,
    WISC1 <-> WISC1, Vd, NA
    WISC2 <-> WISC2, Vd, NA
    WISC3 <-> WISC3, Vd, NA
    WISC4 <-> WISC4, Vd, NA

sem.McArdle <- sem(mod.McArdle, M.McArdle, 204, fixed.x="UNIT", raw=TRUE)
summary(sem.McArdle)

##    Model fit to raw moment matrix.
##
##     Model Chisquare =  83.791   Df =  8 Pr(>Chisq) = 8.4377e-15
##     BIC =  41.246
##
##     Normalized Residuals
##        Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    -0.15300 -0.01840  0.00132 -0.00576  0.02400  0.07760
##
##     Parameter Estimates
##       Estimate Std Error z value Pr(>|z|)
##    B2  8.61354 0.135438  63.5976 0        WISC2 <--- C
##    B3 11.64054 0.168854  68.9387 0        WISC3 <--- C
##    B4 15.40323 0.213071  72.2916 0        WISC4 <--- C
##    Mc  3.01763 0.060690  49.7219 0        C <--- UNIT
##    Vc  0.44343 0.047704   9.2955 0        C <--> C
##    Vd 11.78832 0.674060  17.4885 0        WISC1 <--> WISC1
##
##     Iterations =  37

    ## End(Not run)
```

`specify.model` *Specify a Structural Equation Model*

Description

Create the RAM specification of a structural equation model.

Usage

```
specify.model(file = "")

## S3 method for class 'mod':
print(x, ...)
```

Arguments

file	The (quoted) file from which to read the model specification, including the path to the file if it is not in the current directory. If "" (the default), then the specification is read from the standard input stream, and is terminated by a blank line.
x	An object of class mod to print, as produced by specify.model.
...	Ignored.

Details

Each line of the RAM specification consists of three (unquoted) entries, separated by commas:

1. **Arrow specification:** This is a simple formula, of the form A -> B or, equivalently, B <- A for a regression coefficient (i.e., a single-headed or directional arrow); A <-> A for a variance or A <-> B for a covariance (i.e., a double-headed or bidirectional arrow). Here, A and B are variable names in the model. If a name does not correspond to an observed variable, then it is assumed to be a latent variable. Spaces can appear freely in an arrow specification, and there can be any number of hyphens in the arrows, including zero: Thus, e.g., A->B, A -> B, and A>B are all legitimate and equivalent.
2. **Parameter name:** The name of the regression coefficient, variance, or covariance specified by the arrow. Assigning the same name to two or more arrows results in an equality constraint. Specifying the parameter name as NA produces a fixed parameter.
3. **Value:** start value for a free parameter or value of a fixed parameter. If given as NA, sem will compute the start value.

Lines may end in a comment following #.

See sem for further details on model specification.

Value

An object of class mod, suitable as input for sem.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

See Also

sem

Examples

```
    ## Not run:
model.dhp <- specify.model()
    RParAsp  -> RGenAsp, gam11,  NA
    RIQ      -> RGenAsp, gam12,  NA
    RSES     -> RGenAsp, gam13,  NA
    FSES     -> RGenAsp, gam14,  NA
    RSES     -> FGenAsp, gam23,  NA
    FSES     -> FGenAsp, gam24,  NA
    FIQ      -> FGenAsp, gam25,  NA
    FParAsp  -> FGenAsp, gam26,  NA
    FGenAsp  -> RGenAsp, beta12, NA
    RGenAsp  -> FGenAsp, beta21, NA
    RGenAsp  -> ROccAsp,  NA,     1
    RGenAsp  -> REdAsp,  lam21,  NA
    FGenAsp  -> FOccAsp,  NA,     1
    FGenAsp  -> FEdAsp,  lam42,  NA
    RGenAsp <-> RGenAsp, ps11,   NA
    FGenAsp <-> FGenAsp, ps22,   NA
    RGenAsp <-> FGenAsp, ps12,   NA
    ROccAsp <-> ROccAsp, theta1, NA
    REdAsp  <-> REdAsp,  theta2, NA
    FOccAsp <-> FOccAsp, theta3, NA
    FEdAsp  <-> FEdAsp,  theta4, NA

model.dhp
    ## End(Not run)
```

---

standardized.coefficients

Standardized Coefficients for Structural Equation Models

Description

These functions calculate standardized regression coefficients for structural equation models. The function `std.coef` is simply an abbreviation for `standardized.coefficients`.

Usage

```
standardized.coefficients(object, digits=5)

std.coef(...)
```

Arguments

`object`	an object of class `sem` returned by the `sem` function.
`digits`	number of digits for printed output.
`...`	arguments to pass to `standardized.coefficients`.

Value

Returns a data frame with the standardized coefficients, labelled both by parameter names and by arrows in the path diagram for the model.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

References

Bollen, K. A. (1989) *Structural Equations With Latent Variables.* Wiley.

See Also

sem

Examples

```
# ------------- assumes that Duncan, Haller and Portes peer-influences model
# -------------     has been fit and is in sem.dhp.1
## Not run:
standardized.coefficients(sem.dhp.1)

##           Std. Estimate
## 1  gam11  0.210278      RGenAsp <--- RParAsp
## 2  gam12  0.325612      RGenAsp <--- RIQ
## 3  gam13  0.284855      RGenAsp <--- RSES
## 4  gam14  0.093702      RGenAsp <--- FSES
## 5  gam23  0.074576      FGenAsp <--- RSES
## 6  gam24  0.275763      FGenAsp <--- FSES
## 7  gam25  0.420558      FGenAsp <--- FIQ
## 8  gam26  0.192224      FGenAsp <--- FParAsp
## 9  beta12 0.199418      RGenAsp <--- FGenAsp
## 10 beta21 0.217521      FGenAsp <--- RGenAsp
## 11        0.766717      ROccAsp <--- RGenAsp
## 12 lam21  0.814771      REdAsp <--- RGenAsp
## 13        0.829943      FOccAsp <--- FGenAsp
## 14 lam42  0.771619      FEdAsp <--- FGenAsp
    ## End(Not run)
```

---

tsls *Two-Stage Least Squares*

Description

Fits an equation in a structural-equation model by two-stage least squares. This is equivalent to direct instrumental-variables estimation when the number of instruments is equal to the number of predictors.

Usage

```
tsls(y, ...)
## S3 method for class 'formula':
tsls(formula, instruments, data, subset, na.action, contrasts=NULL, ...)
## Default S3 method:
tsls(y, X, Z, names=NULL, ...)

## S3 method for class 'tsls':
print(x, ...)
## S3 method for class 'tsls':
summary(object, digits=4, ...)
## S3 method for class 'tsls':
anova(object, model.2, s2, dfe, ...)

## S3 method for class 'tsls':
fitted(object, ...)
## S3 method for class 'tsls':
residuals(object, ...)
## S3 method for class 'tsls':
coef(object, ...)
## S3 method for class 'tsls':
vcov(object, ...)
```

Arguments

`formula`	model formula for structural equation to be estimated; a regression constant is implied if not explicitly omitted.
`instruments`	one-sided model formula specifying instrumental variables.
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment from which tsls is called.
`subset`	an optional vector specifying a subset of observations to be used in fitting the model.
`na.action`	a function that indicates what should happen when the data contain `NA`s. The default is set by the `na.action` option.
`contrasts`	an optional list. See the `contrasts.arg` of `model.matrix.default`.
`y`	Response-variable vector.
`X`	Matrix of predictors, including a constant (if one is in the model).
`Z`	Matrix of instrumental variables, including a constant (if one is in the model).
`names`	optional character vector of names for the columns of the `X` matrix.
`x, object, model.2`	objects of class `tsls` returned by `tsls.formula`, containing nested models to be compared by an incremental F-test. One model should be nested in the other; the order of models is immaterial.
`s2`	an optional estimate of error variance for the denominator of the F-test. If missing, the error-variance estimate is taken from the larger model.

dfe	optional error degrees of freedom, to be specified when an estimate of error variance is given.
digits	number of digits for summary output.
...	arguments to be passed down.

Value

tsls.formula returns an object of class tsls, with the following components:

n	number of observations.
p	number of parameters.
coefficients	parameter estimates.
V	estimated covariance matrix of coefficients.
s	residual standard error.
residuals	vector of residuals.
response	vector of response values.
X	model matrix.
Z	instrumental-variables matrix.
response.name	name of response variable, or expression evaluating to response.
formula	model formula.
instruments	one-sided formula for instrumental variables.

Author(s)

John Fox ⟨jfox@mcmaster.ca⟩

References

Fox, J. (1979) Simultaneous equation models and two-stage least-squares. In Schuessler, K. F. (ed.) *Sociological Methodology 1979*, Jossey-Bass.

Greene, W. H. (1993) *Econometric Analysis, Second Edition*, Macmillan.

See Also

sem

Examples

```
data(Kmenta)
summary(tsls(Q ~ P + D, ~ D + F + A, data=Kmenta))     # demand equation

##  2SLS Estimates
##
##  Model Formula: Q ~ P + D
##
##  Instruments: ~D + F + A
```

```
##
##  Residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##  -3.43e+00 -1.24e+00 -1.89e-01 -2.49e-13  1.58e+00  2.49e+00
##
##              Estimate Std. Error t value  Pr(>|t|)
##  (Intercept)  94.6333    7.92084  11.947 1.076e-09
##  P            -0.2436    0.09648  -2.524 2.183e-02
##  D             0.3140    0.04694   6.689 3.811e-06
##
##  Residual standard error: 1.9663 on 17 degrees of freedom

summary(tsls(Q ~ P + F + A, ~ D + F + A, data=Kmenta)) # supply equation

##  2SLS Estimates
##
##  Model Formula: Q ~ P + F + A
##
##  Instruments: ~D + F + A
##
##  Residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##  -4.87e+00 -1.26e+00  6.42e-01 -5.64e-12  1.47e+00  3.49e+00
##
##              Estimate Std. Error t value  Pr(>|t|)
##  (Intercept)  49.5324   12.01053   4.124 7.954e-04
##  P             0.2401    0.09993   2.402 2.878e-02
##  F             0.2556    0.04725   5.410 5.785e-05
##  A             0.2529    0.09966   2.538 2.193e-02
##
##  Residual standard error: 2.4576 on 16 degrees of freedom

anova(tsls(Q ~ P + F + A, ~ D + F + A, data=Kmenta),
    tsls(Q ~ 1, ~ D + F + A, data=Kmenta))
##
##  Analysis of Variance
##
##  Model 1:  Q ~ P + F + A   Instruments: ~D + F + A
##  Model 2:  Q ~ 1   Instruments: ~D + F + A
##
##           Res.Df     RSS Df Sum of Sq      F    Pr(>F)
##  Model 1     16  96.633
##  Model 2     19 268.114  3   171.481 9.4643 0.0007834 ***
##  ---
##  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Index