Package ‘psychonetrics’

Type Package
Title Structural Equation Modeling and Confirmatory Network Analysis
Version 0.11
Author Sacha Epskamp
Maintainer Sacha Epskamp <mail@sachaepskamp.com>
Description Multi-group (dynamical) structural equation models in combination with confirmatory network models from cross-sectional, time-series and panel data <doi:10.31234/osf.io/8ha93>. Allows for confirmatory testing and fit as well as exploratory model search.
License GPL-2
LinkingTo Rcpp (>= 0.11.3), RcppArmadillo, pbv, roptim
Depends R (>= 3.5)
Imports methods, qgraph, numDeriv, dplyr, abind, Matrix, lavaan, corpcor, glasso, mgcv, optimx, VCA, pbapply, parallel, magrittr, IsingSampler, tidyr, psych, GA, combinat, rlang
Suggests psychTools, semPlot, graphicalVAR, metaSEM, mvtnorm, ggplot2
ByteCompile true
URL http://psychonetrics.org/
BugReports https://github.com/SachaEpskamp/psychonetrics/issues
StagedInstall true
NeedsCompilation yes
Repository CRAN
Date/Publication 2023-06-05 11:40:02 UTC

R topics documented:

psychonetrics-package .......................................................... 3
bifactor ............................................................................ 5
bootstrap ........................................................................... 6
**R topics documented:**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>changedata</td>
<td>6</td>
</tr>
<tr>
<td>CIplot</td>
<td>7</td>
</tr>
<tr>
<td>compare</td>
<td>9</td>
</tr>
<tr>
<td>covML</td>
<td>10</td>
</tr>
<tr>
<td>dlvm1</td>
<td>11</td>
</tr>
<tr>
<td>duplicationMatrix</td>
<td>17</td>
</tr>
<tr>
<td>emergencystart</td>
<td>18</td>
</tr>
<tr>
<td>esa</td>
<td>18</td>
</tr>
<tr>
<td>factorscores</td>
<td>19</td>
</tr>
<tr>
<td>fit</td>
<td>20</td>
</tr>
<tr>
<td>fixpar</td>
<td>21</td>
</tr>
<tr>
<td>generate</td>
<td>22</td>
</tr>
<tr>
<td>getmatrix</td>
<td>22</td>
</tr>
<tr>
<td>getVCOV</td>
<td>24</td>
</tr>
<tr>
<td>groupequal</td>
<td>24</td>
</tr>
<tr>
<td>Ising</td>
<td>25</td>
</tr>
<tr>
<td>Jonas</td>
<td>28</td>
</tr>
<tr>
<td>latentgrowth</td>
<td>29</td>
</tr>
<tr>
<td>lvm</td>
<td>31</td>
</tr>
<tr>
<td>meta_varcov</td>
<td>42</td>
</tr>
<tr>
<td>MIs</td>
<td>45</td>
</tr>
<tr>
<td>ml_lvm</td>
<td>47</td>
</tr>
<tr>
<td>ml_tsdvm1</td>
<td>51</td>
</tr>
<tr>
<td>modelsearch</td>
<td>52</td>
</tr>
<tr>
<td>parameters</td>
<td>54</td>
</tr>
<tr>
<td>parequal</td>
<td>55</td>
</tr>
<tr>
<td>partialprune</td>
<td>56</td>
</tr>
<tr>
<td>prune</td>
<td>57</td>
</tr>
<tr>
<td>psychonetrics-class</td>
<td>58</td>
</tr>
<tr>
<td>psychonetrics_update</td>
<td>60</td>
</tr>
<tr>
<td>runmodel</td>
<td>61</td>
</tr>
<tr>
<td>setestimator</td>
<td>62</td>
</tr>
<tr>
<td>setverbose</td>
<td>63</td>
</tr>
<tr>
<td>simplestructure</td>
<td>64</td>
</tr>
<tr>
<td>StarWars</td>
<td>64</td>
</tr>
<tr>
<td>stepup</td>
<td>65</td>
</tr>
<tr>
<td>tsdlvm1</td>
<td>67</td>
</tr>
<tr>
<td>unionmodel</td>
<td>72</td>
</tr>
<tr>
<td>var1</td>
<td>73</td>
</tr>
<tr>
<td>varcov</td>
<td>78</td>
</tr>
</tbody>
</table>

**Index**

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>84</td>
</tr>
</tbody>
</table>
Description

Multi-group (dynamical) structural equation models in combination with confirmatory network models from cross-sectional, time-series and panel data <doi:10.31234/osf.io/8ha93>. Allows for confirmatory testing and fit as well as exploratory model search.

Details

The DESCRIPTION file:

Package: psychonetrics
Type: Package
Title: Structural Equation Modeling and Confirmatory Network Analysis
Version: 0.11
Author: Sacha Epskamp
Maintainer: Sacha Epskamp <mail@sachaepskamp.com>
Description: Multi-group (dynamical) structural equation models in combination with confirmatory network models from cross-sectional, time-series and panel data <doi:10.31234/osf.io/8ha93>. Allows for confirmatory testing and fit as well as exploratory model search.
License: GPL-2
LinkingTo: Rcpp (>= 0.11.3), RcppArmadillo, pbv, roptim
Depends: R (>= 3.5)
Imports: methods, qgraph, numDeriv, dplyr, abind, Matrix, lavaan, corpcor, glasso, mgcv, optimx, VCA, pbapply, parallel, magrittr, IsingSampler, tidyr, psych, GA, combinat, rlang
Suggests: psychTools, semPlot, graphicalVAR, metaSEM, mvtnorm, ggplot2
ByteCompile: true
URL: http://psychonetrics.org/
BugReports: https://github.com/SachaEpskamp/psychonetrics/issues
StagedInstall: true

Index of help topics:

CIplot Plot Analytic Confidence Intervals
Ising Ising model
Jonas Jonas dataset
MIs Print modification indices
StarWars Star Wars dataset
addMIs Model updating functions
bifactor Bi-factor models
bootstrap Bootstrap a psychonetrics model
changedata Change the data of a psychonetrics object
compare Model comparison
covML Maximum likelihood covariance estimate
dlvm1 Lag-1 dynamic latent variable model family of psychonetrics models for panel data
duplicationMatrix Model matrices used in derivatives
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>emergencystart</td>
<td>Reset starting values to simple defaults</td>
</tr>
<tr>
<td>esa</td>
<td>Ergodic Subspace Analysis</td>
</tr>
<tr>
<td>factorscores</td>
<td>Compute factor scores</td>
</tr>
<tr>
<td>fit</td>
<td>Print fit indices</td>
</tr>
<tr>
<td>fixpar</td>
<td>Parameters modification</td>
</tr>
<tr>
<td>generate</td>
<td>Generate data from a fitted psychonetrics object</td>
</tr>
<tr>
<td>getVCOV</td>
<td>Obtain the asymptotic covariance matrix</td>
</tr>
<tr>
<td>getmatrix</td>
<td>Extract an estimated matrix</td>
</tr>
<tr>
<td>groupequal</td>
<td>Group equality constrains</td>
</tr>
<tr>
<td>latentgrowth</td>
<td>Latnet growth curve model</td>
</tr>
<tr>
<td>lvm</td>
<td>Continuous latent variable family of psychonetrics models</td>
</tr>
<tr>
<td>meta_varcov</td>
<td>Variance-covariance and GGM meta analysis</td>
</tr>
<tr>
<td>ml_lvm</td>
<td>Multi-level latent variable model family</td>
</tr>
<tr>
<td>ml_tsdlvm1</td>
<td>Multi-level Lag-1 dynamic latent variable model family of psychonetrics models for time-series data</td>
</tr>
<tr>
<td>modelsearch</td>
<td>Stepwise model search</td>
</tr>
<tr>
<td>parameters</td>
<td>Print parameter estimates</td>
</tr>
<tr>
<td>parequal</td>
<td>Set equality constrains across parameters</td>
</tr>
<tr>
<td>partialprune</td>
<td>Partial pruning of multi-group models</td>
</tr>
<tr>
<td>prune</td>
<td>Stepdown model search by pruning non-significant parameters.</td>
</tr>
<tr>
<td>psychonetrics-class</td>
<td>Class ’&quot;psychonetrics&quot;'</td>
</tr>
<tr>
<td>psychonetrics-package</td>
<td>Structural Equation Modeling and Confirmatory Network Analysis</td>
</tr>
<tr>
<td>runmodel</td>
<td>Run a psychonetrics model</td>
</tr>
<tr>
<td>setestimator</td>
<td>Convenience functions</td>
</tr>
<tr>
<td>setverbose</td>
<td>Should messages of computation progress be printed?</td>
</tr>
<tr>
<td>simplestructure</td>
<td>Generate factor loadings matrix with simple structure</td>
</tr>
<tr>
<td>stepup</td>
<td>Stepup model search along modification indices</td>
</tr>
<tr>
<td>tsdlvm1</td>
<td>Lag-1 dynamic latent variable model family of psychonetrics models for time-series data</td>
</tr>
<tr>
<td>unionmodel</td>
<td>Unify models across groups</td>
</tr>
<tr>
<td>var1</td>
<td>Lag-1 vector autoregression family of psychonetrics models</td>
</tr>
<tr>
<td>varcov</td>
<td>Variance-covariance family of psychonetrics models</td>
</tr>
</tbody>
</table>

This package can be used to perform Structural Equation Modeling and confirmatory network modeling. Current implemented families of models are (1) the variance–covariance matrix (varcov), (2) the latent variable model (lvm), (3) the lag-1 vector autoregression model (var1), and (4) the dynamical lag-1 latent variable model for panel data (dlvm1) and for time-series data (tsdlvm1).
bifactor

Author(s)

Sacha Epskamp
Maintainer: Sacha Epskamp <mail@sachaepskamp.com>

References

More information: psychonetrics.org

---

bifactor \hspace{1cm} Bi-factor models

---

Description

Wrapper to \texttt{lvm} to specify a bi-factor model.

Usage

\begin{verbatim}
bifactor(data, lambda, latents, bifactor = "g", ...)
\end{verbatim}

Arguments

\begin{verbatim}
data \hspace{1cm} The data as used by \texttt{lvm}
lambda \hspace{1cm} The factor loadings matrix *without* the bifactor, as used by \texttt{lvm}
latents \hspace{1cm} A vector of names of the latent variables, as used by \texttt{lvm}
bifactor \hspace{1cm} Name of the bifactor
... \hspace{1cm} Arguments sent to \texttt{lvm}
\end{verbatim}

Value

An object of the class psychonetrics (psychonetrics-class)

Author(s)

Sacha Epskamp
bootstrap  

*Bootstrap a psychonetrics model*

**Description**

This function will bootstrap the data (once) and return a new unevaluated psychonetrics object. It requires storedata = TRUE to be used when forming a model.

**Usage**

```r
bootstrap(x, replacement = TRUE, proportion = 1, verbose = TRUE, storedata = FALSE, baseline_saturated = TRUE)
```

**Arguments**

- `x` A psychonetrics model.
- `replacement` Logical, should new samples be drawn with replacement?
- `proportion` Proportion of sample to be drawn. Set to lower than $1$ for subsampling.
- `verbose` Logical, should messages be printed?
- `storedata` Logical, should the bootstrapped data also be stored?
- `baseline_saturated` Logical, should the baseline and saturated models be included?

**Value**

An object of the class psychonetrics (*psychonetrics-class*)

**Author(s)**

Sacha Epskamp

---

changedata  

*Change the data of a psychonetrics object*

**Description**

This function can be used to change the data in a psychonetrics object.

**Usage**

```r
changedata(x, data, covs, nobs, means, groups, missing = "listwise")
```
CIplot

**Arguments**

- `x` A psychonetrics model.
- `data` A data frame encoding the data used in the analysis. Can be missing if `covs` and `nobs` are supplied.
- `covs` A sample variance–covariance matrix, or a list/array of such matrices for multiple groups. IMPORTANT NOTE: psychonetrics expects the maximum likelihood (ML) covariance matrix, which is NOT obtained from `cov` directly. Manually rescale the result of `cov` with \((nobs - 1)/nobs\) to obtain the ML covariance matrix.
- `nobs` The number of observations used in `covs` and `means`, or a vector of such numbers of observations for multiple groups.
- `means` A vector of sample means, or a list/matrix containing such vectors for multiple groups.
- `groups` An optional string indicating the name of the group variable in `data`.
- `missing` How should missingness be handled in computing the sample covariances and number of observations when `data` is used. Can be "listwise" for listwise deletion, or "pairwise" for pairwise deletion.

**Value**

An object of the class psychonetrics (`psychonetrics-class`)

**Author(s)**

Sacha Epskamp

---

**Description**

Function to plot analytic confidence intervals (CI) of matrix elements estimated in psychonetrics.

**Usage**

```
CIplot(x, matrices, alpha_ci = 0.05,
       alpha_color = c(0.05, 0.01, 0.001, 1e-04),
       labels, labels2, labelstart, print = TRUE,
       major_break = 0.2, minor_break = 0.1)
```
Arguments

- `x`: A psychonetrics model.
- `matrices`: Vector of strings indicating the matrices to plot CIs for.
- `alpha_ci`: The alpha level used for the CIs.
- `alpha_color`: A vector of alphas used for coloring the CIs.
- `labels`: The labels for the variables associated with the rows of a matrix.
- `labels2`: The labels for the variables associated with the columns of a matrix. Defaults to the value of `labels` for square matrices.
- `labelstart`: The value to determine if labels are printed to the right or to the left of the CI.
- `print`: Logical, should the plots also be printed? Only works when one matrix is used in `matrices`.
- `major_break`: Numeric indicating the step size between major breaks.
- `minor_break`: Numeric indicating the step size between minor breaks.

Value

A single ggplot2 object, or a list of ggplot2 objects for each matrix requested.

Author(s)

Sacha Epskamp

Examples

```r
### Example from ?ggm ###
# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")

# Let's take the agreeableness items, and gender:
ConsData <- bfi %>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML")

# Define variables:
vars <- names(ConsData)[1:5]

# Let's fit an empty GGM:
mod0 <- ggm(ConsData, vars = vars)

# Run the model:
mod0 <- mod0 %>% runmodel

# Labels:
labels <- c(
```
"indifferent to the feelings of others",
"inquire about others' well-being",
"comfort others",
"love children",
"make people feel at ease")

# Plot the CIs:
CIplot(mod0, "omega", labels = labels, labelstart = 0.2)

### Example from ?gvar ###
library("dplyr")
library("graphicalVAR")

beta <- matrix(c(
  0,0.5,
  0.5,0
),2,2,byrow=TRUE)
kappa <- diag(2)
simData <- graphicalVARsim(50, beta, kappa)

# Form model:
model <- gvar(simData)

# Evaluate model:
model <- model %>% runmodel

# Plot the CIs:
CIplot(model, "beta")

---

**compare**

**Model comparison**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This function will print a table comparing multiple models on chi-square, AIC and BIC.</td>
</tr>
</tbody>
</table>

**Usage**

```r
compare(...)```

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

**Arguments**

- `...` Any number of psychonetrics models. Can be named to change the rownames of the output.
- `x` Output of the compare function.
Value
A data frame with chi-square values, degrees of freedoms, RMSEAs, AICs, and BICs.

Author(s)
Sacha Epskamp

covML
Maximum likelihood covariance estimate

Description
These functions complement the base R cov function by simplifying obtaining maximum likelihood (ML) covariance estimates (denominator n) instead of unbiased (UB) covariance estimates (denominator n-1). The function covML can be used to obtain ML estimates, the function covUBtoML transforms from UB to ML estimates, and the function covMLtoUB transforms from UB to ML estimates.

Usage
covML(x, ...)
covUBtoML(x, n, ...)
covMLtoUB(x, n, ...)

Arguments
x A dataset
n The sample size
... Arguments sent to the cov function.

Author(s)
Sacha Epskamp <mail@sachaepskamp.com>

Examples
data("StarWars")
Y <- StarWars[,1:10]

# Unbiased estimate:
UB <- cov(Y)

# ML Estimate:
ML <- covML(Y)

# Check:
all(abs(UB - covMLtoUB(ML, nrow(Y))) < sqrt(.Machine$double.eps))
all(abs(ML - covUBtoML(UB, nrow(Y))) < sqrt(.Machine$double.eps))
**dlvm1**  
*Lag-1 dynamic latent variable model family of psychonometrics models for panel data*

**Description**

This is the family of models that models a dynamic factor model on panel data. There are four covariance structures that can be modeled in different ways: `within_latent`, `between_latent` for the within-person and between-person latent (contemporaneous) models respectively, and `within_residual`, `between_residual` for the within-person and between-person residual models respectively. The `panelgvar` wrapper function sets the `lambda` to an identity matrix, all residual variances to zero, and models within-person and between-person latent (contemporaneous) models as GGMs. The `panelvar` wrapper does the same but models contemporaneous relations as a variance-covariance matrix. Finally, the `panel_lvgvar` wrapper automatically models all latent networks as GGMs.

**Usage**

```r
dlvm1(data, vars, lambda, within_latent = c("cov", "chol", "prec", "ggm"), within_residual = c("cov", "chol", "prec", "ggm"), between_latent = c("cov", "chol", "prec", "ggm"), between_residual = c("cov", "chol", "prec", "ggm"), beta = "full", omega_zeta_within = "full", delta_zeta_within = "full", kappa_zeta_within = "full", sigma_zeta_within = "full", lowertri_zeta_within = "full", omega_epsilon_within = "empty", delta_epsilon_within = "empty", kappa_epsilon_within = "empty", sigma_epsilon_within = "empty", lowertri_epsilon_within = "empty", omega_zeta_between = "full", delta_zeta_between = "full", kappa_zeta_between = "full", sigma_zeta_between = "full", lowertri_zeta_between = "full", omega_epsilon_between = "empty", delta_epsilon_between = "empty", kappa_epsilon_between = "empty", sigma_epsilon_between = "empty", lowertri_epsilon_between = "empty", nu, mu_eta, identify = TRUE, identification = c("loadings", "variance"), latents, groups, covs, means, nobs, covtype = c("choose", "ML", "UB"), missing = "listwise", equal = "none", baseline_saturated = TRUE, estimator = "ML", optimizer, storedata = FALSE, verbose = FALSE, sampleStats)
```

```r
panelgvar(data, vars, within_latent = c("ggm","chol","cov","prec"), between_latent = c("ggm","chol","cov","prec"), ...)
```

```r
panelvar(data, vars, within_latent = c("cov","chol","prec","ggm"), between_latent = c("cov","chol","prec","ggm"), ...)
```
panel_lvgvar(...)  

**Arguments**  

**data**  
A data frame encoding the data used in the analysis. Can be missing if covs and nobs are supplied.  

**vars**  
Required argument. Different from in other psychonetrics models, this must be a *matrix* with each row indicating a variable and each column indicating a measurement. The matrix must be filled with names of the variables in the dataset corresponding to variable i at wave j. NAs can be used to indicate missing waves. The rownames of this matrix will be used as variable names.  

**lambda**  
Required argument. A model matrix encoding the factor loading structure. Each row indicates an indicator and each column a latent. A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.  

**within_latent**  
The type of within-person latent contemporaneous model to be used.  

**within_residual**  
The type of within-person residual model to be used.  

**between_latent**  
The type of between-person latent model to be used.  

**between_residual**  
The type of between-person residual model to be used.  

**beta**  
A model matrix encoding the temporal relationships (transpose of temporal network). A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix. Can also be "full" for a full temporal network or "empty" for an empty temporal network.  

**omega_zeta_within**  
Only used when within_latent = "gmm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.  

**delta_zeta_within**  
Only used when within_latent = "gmm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.  

**kappa_zeta_within**  
Only used when within_latent = "prec". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
sigma_zeta_within
Only used when within_latent = "cov". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

lowertri_zeta_within
Only used when within_latent = "chol". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

omega_epsilon_within
Only used when within_residual = "ggm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

delta_epsilon_within
Only used when within_residual = "ggm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

kappa_epsilon_within
Only used when within_residual = "prec". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

sigma_epsilon_within
Only used when within_residual = "cov". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

lowertri_epsilon_within
Only used when within_residual = "chol". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

omega_zeta_between
Only used when between_latent = "ggm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
delta_zeta_between
Only used when between_latent = "ggm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

kappa_zeta_between
Only used when between_latent = "prec". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

sigma_zeta_between
Only used when between_latent = "cov". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

lowertri_zeta_between
Only used when between_latent = "chol". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

omega_epsilon_between
Only used when between_residuel = "ggm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

delta_epsilon_between
Only used when between_residuel = "ggm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

kappa_epsilon_between
Only used when between_residuel = "prec". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

sigma_epsilon_between
Only used when between_residuel = "cov". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
lowertri_epsilon_between

Only used when between_residual = "chol". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

nu

Optional vector encoding the intercepts of the observed variables. Set elements to 0 to indicate fixed to zero constrains, 1 to indicate free intercepts, and higher integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such a vector.

mu_eta

Optional vector encoding the means of the latent variables. Set elements to 0 to indicate fixed to zero constrains, 1 to indicate free intercepts, and higher integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such a vector.

identify

Logical, should the model be automatically identified?

identification

Type of identification used. "loadings" to fix the first factor loadings to 1, and "variance" to fix the diagonal of the latent variable model matrix (sigma_zeta, lowertri_zeta, delta_zeta or kappa_zeta) to 1.

latents

An optional character vector with names of the latent variables.

groups

An optional string indicating the name of the group variable in data.

covs

A sample variance–covariance matrix, or a list/array of such matrices for multiple groups. IMPORTANT NOTE: psychonetrics expects the maximum likelihood (ML) covariance matrix, which is NOT obtained from cov directly. Manually rescale the result of cov with (nobs - 1)/nobs to obtain the ML covariance matrix.

means

A vector of sample means, or a list/matrix containing such vectors for multiple groups.

nobs

The number of observations used in covs and means, or a vector of such numbers of observations for multiple groups.

missing

How should missingness be handled in computing the sample covariances and number of observations when data is used. Can be "listwise" for listwise deletion, or "pairwise" for pairwise deletion.

equal

A character vector indicating which matrices should be constrained equal across groups.

baseline_saturated

A logical indicating if the baseline and saturated model should be included. Mostly used internally and NOT Recommended to be used manually.

estimator

The estimator to be used. Currently implemented are "ML" for maximum likelihood estimation, "FIML" for full-information maximum likelihood estimation, "ULS" for unweighted least squares estimation, "WLS" for weighted least squares estimation, and "DWLS" for diagonally weighted least squares estimation.

optimizer

The optimizer to be used. Can be one of "nlminb" (the default R nlminb function), "ucminf" (from the optimr package), and C++ based optimizers "cpp_L-BFGS-B", "cpp_BFGS", "cpp_CG", "cpp_SANN", and "cpp_Nelder-Mead". The C++ optimizers are faster but slightly less stable. Defaults to "nlminb".
storedata Logical, should the raw data be stored? Needed for bootstrapping (see `bootstrap`).

verbose Logical, should progress be printed to the console?

sampleStats An optional sample statistics object. Mostly used internally.

covtype If 'covs' is used, this is the type of covariance (maximum likelihood or unbiased) the input covariance matrix represents. Set to "ML" for maximum likelihood estimates (denominator n) and "UB" to unbiased estimates (denominator n-1). The default will try to find the type used, by investigating which is most likely to result from integer valued datasets.

Arguments sent to `dlvm1`.

Value

An object of the class psychonetrics (psychonetrics-class)

Author(s)

Sacha Epskamp

Examples

```
library("dplyr")

# Smoke data cov matrix, based on LISS data panel https://www.dataarchive.lissdata.nl
smoke <- structure(c(47.2361758611759, 43.5366809116809, 41.0057465682466,
                      43.5366809116809, 57.9789886039886, 47.6992521367521,
                      41.0057465682466, 47.6992521367521, 53.0669434731935), .Dim = c(3L, 3L),
                      .Dimnames = list(c("smoke2008", "smoke2009", "smoke2010"), c("smoke2008",
                      "smoke2009", "smoke2010")))

# Design matrix:
design <- matrix(rownames(smoke),1,3)

# Form model:
mod <- panelvar(vars = design, covs = smoke, nobs = 352)

# Run model:
mod <- mod %>% runmodel

# Evaluate fit:
mod %>% fit
```
duplicationMatrix

Model matrices used in derivatives

Description

These matrices are used in the analytic gradients

Usage

duplicationMatrix(n, diag = TRUE)
eliminationMatrix(n, diag = TRUE)
diagonalizationMatrix(n)

Arguments

n : Number of rows and columns in the original matrix
diag : Logical indicating if the diagonal should be included (set to FALSE for derivative of vech(x))

Value

A sparse matrix

Author(s)

Sacha Epskamp

Examples

# Duplication matrix for 10 variables:
duplicationMatrix(10)

# Elimination matrix for 10 variables:
eliminationMatrix(10)

# Diagonalization matrix for 10 variables:
diagonalizationMatrix(10)
emergencystart  
Reset starting values to simple defaults

Description
This function overwrites the starting values to simple defaults. This can help in cases where optimization fails.

Usage
emergencystart(x)

Arguments
  x          A psychonetrics model.

Value
A psychonetrics model.

Author(s)
Sacha Epskamp

esa  
Ergodic Subspace Analysis

Description
These functions implement Ergodic Subspace Analysis by von Oertzen, Schmiedek and Voelkle (2020). The functions can be used on the output of a dlvm1 model, or manually by supplying a within persons and between persons variance-covariance matrix.

Usage
esa(x, cutoff = 0.1,  
  between = c("crosssection", "between"))
esa_manual(sigma_wp, sigma_bp, cutoff = 0.1)  
## S3 method for class 'esa'
print(x, printref = TRUE, ...)
## S3 method for class 'esa_manual'
print(x, printref = TRUE, ...)
## S3 method for class 'esa'
plot(x, plot = c("observed", "latent"), ...)
## S3 method for class 'esa_manual'
plot(x, ...)
**factorscores**

Compute factor scores

---

**Description**

Currently, only the lvm framework with single group and no missing data is supported.

**Usage**

```r
factorscores(data, model, method = c("bartlett", "regression"))
```
Arguments

- **data**: Dataset to compute factor scores for
- **model**: A psychonetrics model
- **method**: The method to use: "regression" or "bartlett"

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

---

fit

*Print fit indices*

Description

This function will print all fit indices of the model.

Usage

```r
fit(x)
```

Arguments

- **x**: A psychonetrics model.

Value

Invisibly returns a data frame with fit measure estimates.

Author(s)

Sacha Epskamp

Examples

```r
# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")

# Let's take the agreeableness items, and gender:
ConsData <- bfi %>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML"

# Define variables:
vars <- names(ConsData)[1:5]
```
# Let's fit an empty GGM:
mod0 <- ggm(ConsData, vars = vars, omega = "empty")

# Run model:
mod0 <- mod0 %>% runmodel

# Inspect fit:
mod0 %>% fit # Pretty bad fit...

---

## fixpar

### Parameters modification

#### Description

The fixpar function can be used to fix a parameter to some value (Typically zero), and the freepar function can be used to free a parameter from being fixed to a value.

#### Usage

```r
fixpar(x, matrix, row, col, value = 0, group, verbose, log = TRUE, runmodel = FALSE, ...)
freepar(x, matrix, row, col, start, group, verbose, log = TRUE, runmodel = FALSE, startEPC = TRUE, ...)
```

#### Arguments

- **x**: A psychonetrics model.
- **matrix**: String indicating the matrix of the parameter
- **row**: Integer or string indicating the row of the matrix of the parameter
- **col**: Integer or string indicating the column of the matrix of the parameter
- **value**: Used in fixpar to indicate the value to which a parameters is constrained
- **start**: Used in freepar to indicate the starting value of the parameter
- **group**: Integer indicating the group of the parameter to be constrained
- **verbose**: Logical, should messages be printed?
- **log**: Logical, should the log be updated?
- **runmodel**: Logical, should the model be updated?
- **startEPC**: Logical, should the starting value be set at the expected parameter change?
- **...**: Arguments sent to runmodel

#### Value

An object of the class psychonetrics (psychonetrics-class)
**generate**

*Generate data from a fitted psychonetrics object*

**Description**

This function will generate new data from the estimated mean and variance-covariance structure of a psychonetrics model.

**Usage**

```r
generate(x, n = 500)
```

**Arguments**

- `x`: A psychonetrics model.
- `n`: Number of cases to sample per group.

**Value**

A data frame with simulated data

---

**getmatrix**

*Extract an estimated matrix*

**Description**

This function will extract an estimated matrix, and will either return a single matrix for single group models or a list of such matrices for multiple group models.

**Usage**

```r
getmatrix(x, matrix, group, threshold = FALSE, alpha = 0.01, adjust = c("none", "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr"), mode = c("tested", "all"), diag = TRUE)
```
getmatrix

Arguments

- \texttt{x}\hspace{1cm} A psychonetrics model.
- \texttt{matrix}\hspace{1cm} String indicating the matrix to be extracted.
- \texttt{group}\hspace{1cm} Integer indicating the group for the matrix to be extracted.
- \texttt{threshold}\hspace{1cm} Logical. Should the matrix be thresholded (non-significant values set to zero? Can also be a value with an absolute threshold below which parameters are set to zero.)
- \texttt{alpha}\hspace{1cm} Significance level to use.
- \texttt{adjust}\hspace{1cm} p-value adjustment method to use. See \texttt{p.adjust}.
- \texttt{mode}\hspace{1cm} Mode for adjusting for multiple comparisons. Should all parameters be considered as the total number of tests or only the tested parameters (parameters of interest)?
- \texttt{diag}\hspace{1cm} Set to FALSE to set diagonal elements to zero.

Value

A matrix of parameter estimates, or a list of such matrices for multiple group models.

Author(s)

Sacha Epskamp

Examples

```r
# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")

# Let's take the agreeableness items, and gender:
ConsData <- bfi %>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML"

# Define variables:
vars <- names(ConsData)[1:5]

# Let's fit a full GGM:
mod <- ggm(ConsData, vars = vars, omega = "full")

# Run model:
mod <- mod %>% runmodel

# Obtain network:
mod %>% getmatrix("omega")
```
getVCOV

Get the asymptotic covariance matrix

**Description**

This function can be used to obtain the estimated asymptotic covariance matrix from a psychonetrics object.

**Usage**

```r
getVCOV(model)
```

**Arguments**

- `model`: A psychonetrics model.

**Value**

This function returns a matrix.

**Author(s)**

Sacha Epskamp

groupequal

Group equality constrains

**Description**

The groupequal function constrains parameters equal across groups, and the groupfree function frees equality constrains across groups.

**Usage**

```r
groupequal(x, matrix, row, col, verbose, log = TRUE, runmodel = FALSE, identify = TRUE, ...)
groupfree(x, matrix, row, col, verbose, log = TRUE, runmodel = FALSE, identify = TRUE, ...)
```
**Ising**

**Arguments**

- `x` A psychonetrics model.
- `matrix` String indicating the matrix of the parameter
- `row` Integer or string indicating the row of the matrix of the parameter
- `col` Integer or string indicating the column of the matrix of the parameter
- `verbose` Logical, should messages be printed?
- `log` Logical, should the log be updated?
- `runmodel` Logical, should the model be updated?
- `identify` Logical, should the model be identified?
- `...` Arguments sent to `runmodel`

**Value**

An object of the class psychonetrics (psychonetrics-class)

**Author(s)**

Sacha Epskamp

---

**Ising**

Ising model

**Description**

This is the family of Ising models fit to dichotomous datasets. Note that the input matters (see also https://arxiv.org/abs/1811.02916) in this model! Models based on a dataset that is encoded with -1 and 1 are not entirely equivalent to models based on datasets encoded with 0 and 1 (non-equivalences occur in multi-group settings with equality constrains).

**Usage**

```r
Ising(data, omega = "full", tau, beta, vars, groups, covs,
     means, nobs, covtype = c("choose", "ML", "UB"),
     responses, missing = "listwise", equal = "none",
     baseline_saturated = TRUE, estimator = "default",
     optimizer, storedata = FALSE, WLS.W, sampleStats,
     identify = TRUE, verbose = FALSE, maxNodes = 20)
```
Arguments

data
A data frame encoding the data used in the analysis. Can be missing if covs and nobs are supplied.

omega
The network structure. Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions nNode x nNode with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

tau
Optional vector encoding the threshold/intercept structure. Set elements to 0 to indicate fixed to zero constrains, 1 to indicate free intercepts, and higher integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such a vector.

beta
Optional scalar encoding the inverse temperature. 1 indicate free beta parameters, and higher integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such scalers.

vars
An optional character vector encoding the variables used in the analysis. Must equal names of the dataset in data.

groups
An optional character vector encoding the variables used in the analysis. Must equal names of the dataset in data.

covs
A sample variance–covariance matrix, or a list/array of such matrices for multiple groups. Make sure covtype argument is set correctly to the type of covariances used.

means
A vector of sample means, or a list/matrix containing such vectors for multiple groups.

nobs
The number of observations used in covs and means, or a vector of such numbers of observations for multiple groups.

covtype
If 'covs' is used, this is the type of covariance (maximum likelihood or unbiased) the input covariance matrix represents. Set to "ML" for maximum likelihood estimates (denominator n) and "UB" to unbiased estimates (denominator n-1). The default will try to find the type used, by investigating which is most likely to result from integer valued datasets.

responses
A vector of dichotomous responses used (e.g., c(-1, 1) or c(0, 1). Only needed when 'covs' is used.)

missing
How should missingness be handled in computing the sample covariances and number of observations when data is used. Can be "listwise" for listwise deletion, or "pairwise" for pairwise deletion. NOT RECOMMENDED TO BE USED YET IN ISING MODEL.

equal
A character vector indicating which matrices should be constrained equal across groups.

baseline_saturated
A logical indicating if the baseline and saturated model should be included. Mostly used internally and NOT Recommended to be used manually.
estimator  The estimator to be used. Currently implemented are "ML" for maximum like-
lihood estimation, "FIML" for full-information maximum likelihood estimation, 
"ULS" for unweighted least squares estimation, "WLS" for weighted least squares 
estimation, and "DWLS" for diagonally weighted least squares estimation. Only 
ML estimation is currently supported for the Ising model.

optimizer  The optimizer to be used. Can be one of "nlminb" (the default R nlminb 
function), "ucminf" (from the optimr package), and C++ based optimizers 
"cpp_L-BFGS-B", "cpp_BFGS", "cpp_CG", "cpp_SANN", and "cpp_Nelder-Mead". 
The C++ optimizers are faster but slightly less stable. Defaults to "nlminb".

storedata  Logical, should the raw data be stored? Needed for bootstrapping (see bootstrap).

WLS.W  Optional WLS weights matrix. CURRENTLY NOT USED.

sampleStats  An optional sample statistics object. Mostly used internally.

identify  Logical, should the model be identified?

verbose  Logical, should messages be printed?

maxNodes  The maximum number of nodes allowed in the analysis. This function will stop 
with an error if more nodes are used (it is not recommended to set this higher).

Details

The Ising Model takes the following form:

\[ \Pr(Y = y) = \frac{\exp(-\beta H(y; \tau, \Omega))}{Z(\tau, \Omega)} \]

With Hamiltonian:

\[ H(y; \tau, \Omega) = -\sum_{i=1}^{m} \tau_i y_i - \sum_{i=2}^{m} \sum_{j=1}^{i-1} \omega_{ij} y_i y_j. \]

And \( Z \) representing the partition function or normalizing constant.

Value

An object of the class psychonetrics

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

References

Irwing, P., Hughes, D., & Booth, T. (Eds.), The Wiley Handbook of Psychometric Testing, 2 Volume 

Examples

```r
library("dplyr")
data("Jonas")

# Variables to use:
vars <- names(Jonas)[1:10]

# Arranged groups to put unfamiliar group first (beta constrained to 1):
Jonas <- Jonas[order(Jonas$group),]

# Form saturated model:
model1 <- Ising(Jonas, vars = vars, groups = "group")

# Run model:
model1 <- model1 %>% runmodel

# Prune-stepup to find a sparse model:
model1b <- model1 %>% prune(alpha = 0.05) %>% stepup(alpha = 0.05)

# Equal networks:
suppressWarnings(
  model2 <- model1 %>% groupequal("omega") %>% runmodel
)

# Prune-stepup to find a sparse model:
model2b <- model2 %>% prune(alpha = 0.05) %>% stepup(mi = "mi_equal", alpha = 0.05)

# Equal thresholds:
model3 <- model2 %>% groupequal("tau") %>% runmodel

# Prune-stepup to find a sparse model:
model3b <- model3 %>% prune(alpha = 0.05) %>% stepup(mi = "mi_equal", alpha = 0.05)

# Equal beta:
model4 <- model3 %>% groupequal("beta") %>% runmodel

# Prune-stepup to find a sparse model:
model4b <- model4 %>% prune(alpha = 0.05) %>% stepup(mi = "mi_equal", alpha = 0.05)

# Compare all models:
compare(
  `1. all parameters free (dense)` = model1,
  `2. all parameters free (sparse)` = model1b,
  `3. equal networks (dense)` = model2,
  `4. equal networks (sparse)` = model2b,
  `5. equal networks and thresholds (dense)` = model3,
  `6. equal networks and thresholds (sparse)` = model3b,
  `7. all parameters equal (dense)` = model4,
  `8. all parameters equal (sparse)` = model4b
) %>% arrange(BIC)
**Description**

Responses of 10 attitude items towards a researcher named Jonas. Participants were shown three photos of Jonas with the text: "This is Jonas, a researcher from Germany who is now becoming a PhD in Psychology". Subsequently, the participants had to answer 10 yes / no questions starting with "I believe that Jonas...", as well as rate their familiarity with Jonas. The sample consists of people familiar with Jonas and not familiar with Jonas, and allows for testing Attitudinal Entropy Framework <doi:10.1080/1047840X.2018.1537246>.

**Usage**

```r
data("Jonas")
```

**Format**

A data frame with 215 observations on the following 12 variables.

- **scientist** ... is a good scientist
- **jeans** ... Is a person that wears beautiful jeans
- **cares** ... really cares about people like you
- **economics** ... would solve our economic problems
- **hardworking** ... is hardworking
- **honest** ... is honest
- **intouch** ... is in touch with ordinary people
- **knowledgeable** ... is knowledgeable
- **makeupmind** ... can't make up his mind
- **getsthingsdone** ... gets things done
- **familiar** Answers to the question "How familiar are you with Jonas?" (three responses possible)
- **group** The question 'familiar' categorized in two groups ("Knows Jonas" and "Doesn't Know Jonas")

**Examples**

```r
data(Jonas)
```

---

### latentgrowth

*Latent growth curve model*

**Description**

Wrapper to *lvm* to specify a latent growth curve model.

**Usage**

```r
latentgrowth(vars, time = seq_len(ncol(vars)) - 1, covariates =
character(0), covariates_as = c("regression",
"covariance"), ...)
```
Arguments

vars Different from in other psychonetrics models, this must be a *matrix* with each row indicating a variable and each column indicating a measurement. The matrix must be filled with names of the variables in the dataset corresponding to variable i at wave j. NAs can be used to indicate missing waves. The rownames of this matrix will be used as variable names.

time A vector with the encoding of each measurement (e.g., 0, 1, 2, 3).
covariates A vector with strings indicating names of between-person covariate variables in the data
covariates_as Should covariates be included as regressions or actual covariates?

... Arguments sent to lvm

Details

See [https://github.com/SachaEpskamp/SEM-code-examples/tree/master/Latent_growth_examples/psychonetrics](https://github.com/SachaEpskamp/SEM-code-examples/tree/master/Latent_growth_examples/psychonetrics) for examples

Value


Author(s)

Sacha Epskamp

Examples

```r
library("dplyr")

# Smoke data cov matrix, based on LISS data panel [https://www.dataarchive.lissdata.nl](https://www.dataarchive.lissdata.nl)
smoke <- structure(c(47.2361758611759, 43.5366809116809, 41.0057465682466, 43.5366809116809, 57.9789886039886, 47.6992521367521, 41.0057465682466, 47.6992521367521, 53.0669434731935), .Dim = c(3L, 3L),

# Design matrix:
design <- matrix(rownames(smoke),1,3)

# Form model:
mod <- latentgrowth(vars = design,
                     covs = smoke, nobs = 352)
```

## Not run:

# Run model:
lvm

Continuous latent variable family of psychonetrics models

Description

This is the family of models that models the data as a structural equation model (SEM), allowing the latent and residual variance-covariance matrices to be further modeled as networks. The latent and residual arguments can be used to define what latent and residual models are used respectively: "cov" (default) models a variance-covariance matrix directly, "chol" models a Cholesky decomposition, "prec" models a precision matrix, and "ggm" models a Gaussian graphical model (Epskamp, Rhemtulla and Borsboom, 2017). The wrapper lnm() sets latent = "ggm" for the latent network model (LNM), the wrapper rnm() sets residual = "ggm" for the residual network model (RNM), and the wrapper lrnm() combines the LNM and RNM.

Usage

lvm(data, lambda, latent = c("cov", "chol", "prec", "ggm"), residual = c("cov", "chol", "prec", "ggm"), sigma_zeta = "full", kappa_zeta = "full", omega_zeta = "full", lowertri_zeta = "full", delta_zeta = "full", sigma_epsilon = "empty", kappa_epsilon = "empty", omega_epsilon = "empty", lowertri_epsilon = "empty", delta_epsilon = "empty", beta = "empty", nu, nu_eta, identify = TRUE, identification = c("loadings", "variance"), vars, latents, groups, covs, means, nobs, missing = "listwise", equal = "none", baseline_saturated = TRUE, estimator = "ML", optimizer, storedata = FALSE, WLS.W, covtype = c("choose", "ML", "UB"), standardize = c("none", "z", "quantile"), sampleStats, verbose = FALSE, simplelambdastart = FALSE)

lnm(...)

rnm(...)

lrnm(...)
Arguments

data
A data frame encoding the data used in the analysis. Can be missing if covs and nobs are supplied.

lambda
A model matrix encoding the factor loading structure. Each row indicates an indicator and each column a latent. A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

latent
The type of latent model used. See description.

residual
The type of residual model used. See description.

sigma_zeta
Only used when latent = "cov". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

kappa_zeta
Only used when latent = "prec". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

omega_zeta
Only used when latent = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

lowertri_zeta
Only used when latent = "chol". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

delta_zeta
Only used when latent = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

sigma_epsilon
Only used when residual = "cov". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
kappa_epsilon
Only used when residual = "prec". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

omega_epsilon
Only used when residual = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

lowertri_epsilon
Only used when residual = "chol". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

delta_epsilon
Only used when residual = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

beta
A model matrix encoding the structural relations between latent variables. A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

nu
Optional vector encoding the intercepts of the observed variables. Set elements to 0 to indicate fixed to zero constrains, 1 to indicate free intercepts, and higher integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such a vector.

nu_eta
Optional vector encoding the intercepts of the latent variables. Set elements to 0 to indicate fixed to zero constrains, 1 to indicate free intercepts, and higher integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such a vector.

identify
Logical, should the model be automatically identified?

identification
Type of identification used. "loadings" to fix the first factor loadings to 1, and "variance" to fix the diagonal of the latent variable model matrix (sigma_zeta, lowertri_zeta, delta_zeta or kappa_zeta) to 1.

vars
An optional character vector encoding the variables used in the analysis. Must equal names of the dataset in data.

latents
An optional character vector with names of the latent variables.

groups
An optional string indicating the name of the group variable in data.
covs  A sample variance–covariance matrix, or a list/array of such matrices for multiple groups. Make sure covtype argument is set correctly to the type of covariances used.

means  A vector of sample means, or a list/matrix containing such vectors for multiple groups.

nobs  The number of observations used in covs and means, or a vector of such numbers of observations for multiple groups.

missing  How should missingness be handled in computing the sample covariances and number of observations when data is used. Can be "listwise" for listwise deletion, or "pairwise" for pairwise deletion.

equal  A character vector indicating which matrices should be constrained equal across groups.

baseline_saturated  A logical indicating if the baseline and saturated model should be included. Mostly used internally and NOT Recommended to be used manually.

estimator  The estimator to be used. Currently implemented are "ML" for maximum likelihood estimation, "FIML" for full-information maximum likelihood estimation, "ULS" for unweighted least squares estimation, "WLS" for weighted least squares estimation, and "DWLS" for diagonally weighted least squares estimation.

optimizer  The optimizer to be used. Can be one of "nlminb" (the default R nlminb function), "ucminf" (from the optimr package), and C++ based optimizers "cpp_L-BFGS-B", "cpp_BFGS", "cpp.CG", "cpp_SANN", and "cpp_Nelder-Mead". The C++ optimizers are faster but slightly less stable. Defaults to "nlminb".

storedata  Logical, should the raw data be stored? Needed for bootstrapping (see bootstrap).

verbose  Logical, should progress be printed to the console?

WLS.W  The weights matrix used in WLS estimation (experimental)

sampleStats  An optional sample statistics object. Mostly used internally.

covtype  If `covs` is used, this is the type of covariance (maximum likelihood or unbiased) the input covariance matrix represents. Set to "ML" for maximum likelihood estimates (denominator n) and "UB" to unbiased estimates (denominator n-1). The default will try to find the type used, by investigating which is most likely to result from integer valued datasets.

standardize  Which standardization method should be used? "none" (default) for no standardization, "z" for z-scores, and "quantile" for a non-parametric transformation to the quantiles of the marginal standard normal distribution.

simplelambdastart  Logical, should simple start values be used for lambda? Setting this to TRUE can avoid some estimation problems.

...  Arguments sent to varcov

Details

The model used in this family is:

$$\text{var}(y) = \Lambda (I - B)^{-1} \Sigma_c (I - B)^{-1\top} \Lambda^\top + \Sigma_c$$
$$\mathcal{E}(y) = \nu + \Lambda(I - B)^{-1}\nu_{et}a$$

in which the latent covariance matrix can further be modeled in three ways. With latent = "chol" as Cholesky decomposition:

$$\Sigma_\zeta = L_\zeta L_\zeta,$$

with latent = "prec" as Precision matrix:

$$\Sigma_\zeta = K_\zeta^{-1},$$

and finally with latent = "ggm" as Gaussian graphical model:

$$\Sigma_\zeta = \Delta_\zeta(I - \Omega_\zeta)^{(1)} \Delta_\zeta.$$  

Likewise, the residual covariance matrix can also further be modeled in three ways. With residual = "chol" as Cholesky decomposition:

$$\Sigma_\epsilon = L_\epsilon L_\epsilon,$$

with latent = "prec" as Precision matrix:

$$\Sigma_\epsilon = K_\epsilon^{-1},$$

and finally with latent = "ggm" as Gaussian graphical model:

$$\Sigma_\epsilon = \Delta_\epsilon(I - \Omega_\epsilon)^{(1)} \Delta_\epsilon.$$  

**Value**

An object of the class psychonetrics (psychonetrics-class)

**Author(s)**

Sacha Epskamp

**References**


**Examples**

library("dplyr")

### Confirmatory Factor Analysis ###

# Example also shown in https://youtu.be/Hdu5z-fwuk8

# Load data:
data(StarWars)

# Originals only:
Lambda <- matrix(1,4)

# Model:
mod0 <- lvm(StarWars, lambda = Lambda, vars = c("Q1","Q5","Q6","Q7"),
identification = "variance", latents = "Originals")
```r
# Run model:
mod0 <- mod0 %>% runmodel

# Evaluate fit:
mod0 %>% fit

# Full analysis
# Factor loadings matrix:
Lambda <- matrix(0, 10, 3)
Lambda[1:4,1] <- 1
Lambda[c(1,5:7),2] <- 1
Lambda[c(1,8:10),3] <- 1

# Observed variables:
obsvars <- paste0("Q",1:10)

# Latents:
latents <- c("Prequels","Original","Sequels")

# Make model:
mod1 <- lvm(StarWars, lambda = Lambda, vars = obsvars,
identification = "variance", latents = latents)

# Run model:
mod1 <- mod1 %>% runmodel

# Look at fit:
mod1

# Look at parameter estimates:
mod1 %>% parameters

# Look at modification indices:
mod1 %>% MIs

# Add and refit:
mod2 <- mod1 %>% freepar("sigma_epsilon","Q10","Q4") %>% runmodel

# Compare:
compare(original = mod1, adjusted = mod2)

# Fit measures:
mod2 %>% fit

### Path diagrams ###
# semPlot is not (yet) supported by default, but can be used as follows:
# Load packages:
library("semPlot")

# Estimates:
lambdaEst <- getmatrix(mod2, "lambda")
psiEst <- getmatrix(mod2, "sigma_zeta")
```
thetaEst <- getmatrix(mod2, "sigma_epsilon")

# LISREL Model: LY = Lambda (lambda-y), TE = Theta (theta-epsilon), PS = Psi
mod <- lisrelModel(LY = lambdaEst, PS = psiEst, TE = thetaEst)

# Plot with semPlot:
semPaths(mod, "std", "est", as.expression = "nodes")

# We can make this nicer (set whatLabels = "none" to hide labels):
semPaths(mod,

# this argument controls what the color of edges represent. In this case,
# standardized parameters:
  what = "std",

# This argument controls what the edge labels represent. In this case, parameter
# estimates:
  whatLabels = "est",

# This argument draws the node and edge labels as mathematical expressions:
  as.expression = "nodes",

# This will plot residuals as arrows, closer to what we use in class:
  style = "lisrel",

# This makes the residuals larger:
  residScale = 10,

# qgraph colorblind friendly theme:
  theme = "colorblind",

# tree layout options are "tree", "tree2", and "tree3":
  layout = "tree2",

# This makes the latent covariances connect at a cardinal center point:
  cardinal = "lat cov",

# Changes curve into rounded straight lines:
  curvePivot = TRUE,

# Size of manifest variables:
  sizeMan = 4,

# Size of latent variables:
  sizeLat = 10,

# Size of edge labels:
  edge.label.cex = 1,

# Sets the margins:
  mar = c(9,1,8,1),
# Prevents re-ordering of observed variables:
reorder = FALSE,

# Width of the plot:
width = 8,

# Height of plot:
height = 5,

# Colors according to latents:
groups = "latents",

# Pastel colors:
pastel = TRUE,

# Disable borders:
borders = FALSE

# Use arguments filetype = "pdf" and filename = "semPlotExample1" to store PDF

### Latent Network Modeling ###

# Latent network model:
lnm <- lvm(StarWars, lambda = Lambda, vars = obsvars,
        latents = latents, identification = "variance",
        latent = "ggm")

# Run model:
lnm <- lnm %>% runmodel

# Look at parameters:
lnm %>% parameters

# Remove non-sig latent edge:
lnm <- lnm %>% prune(alpha = 0.05)

# Compare to the original CFA model:
compare(cfa = mod1, lnm = lnm)

# Plot network:
library("qgraph")
qgraph(lnm@modelmatrices[[1]]$omega_zeta, labels = latents,
        theme = "colorblind", vsize = 10)

# A wrapper for the latent network model is the lnm function:
lnm2 <- lnm(StarWars, lambda = Lambda, vars = obsvars,
        latents = latents, identification = "variance")
lnm2 <- lnm2 %>% runmodel %>% prune(alpha = 0.05)
compare(lnm, lnm2) # Is the same as the model before.

# I could also estimate a "residual network model", which adds partial correlations to
# the residual level:
# This can be done using lvm(..., residual = "ggm") or with rnm(...)

rnmm <- rnm(StarWars, lambda = Lambda, vars = obsvars,
            latents = latents, identification = "variance")

# Stepup search:
rnmm <- rnmm %>% stepup

# It will estimate the same model (with link Q10 - Q4) as above. In the case of only one
# partial correlation, There is no difference between residual covariances (SEM) or
# residual partial correlations (RNM).

# For more information on latent and residual network models, see:
# Epskamp, S., Rhemtulla, M.T., & Borsboom, D. Generalized Network Psychometrics:
# Combining Network and Latent Variable Models
# (2017). Psychometrika. doi:10.1007/s11336-017-9557-x

### Gaussian graphical models ###

# All psychonetrics functions (e.g., lvm, lnm, rnm...) allow input via a covariance
# matrix, with the "covs" and "nobs" arguments.
# The following fits a baseline GGM network with no edges:
S <- (nrow(StarWars) - 1)/(nrow(StarWars)) * cov(StarWars[,1:10])
ggmmod <- ggm(covs = S, nobs = nrow(StarWars))

# Run model with stepup search and pruning:
ggmmod <- ggmmod %>% prune %>% modelsearch

# Fit measures:
ggmmod %>% fit

# Plot network:
nodeNames <- c(
    "I am a huge Star Wars\nfan! (star what?)",
    "I would trust this person\nwith my democracy.",
    "I enjoyed the story of\nAnakin's early life.",
    "The special effects in\nthis scene are awful (Battle of\nGeonosis).",
    "I would trust this person\nwith my life.",
    "I found Darth Vader's big\nreveal in 'Empire' one of the greatest
moments in movie history.",
    "The special effects in\nthis scene are amazing (Death Star\nExplosion).",
    "If possible, I would\ndefinitely buy this\ndroid.",
    "The story in the Star\nWars sequels is an improvement to\nthe previous movies.",
    "The special effects in\nthis scene are marvellous (Starkiller\nBase Firing).
"
)
library("qgraph")
qgraph(as.matrix(ggmmod@modelmatrices[[1]]$omega), nodeNames = nodeNames,
       legend.cex = 0.25, theme = "colorblind", layout = "spring")

# We can actually compare this model statistically (note they are not nested) to the
# latent variable model:
compare(original_cfa = mod1, adjusted_cfa = mod2, exploratory_ggm = ggmmod)
### Measurement invariance ###

Let's say we are interested in seeing if people >= 30 like the original trilogy better than people < 30.

First we can make a grouping variable:
```
StarWars$agegroup <- ifelse(StarWars$Q12 < 30, "young", "less young")
```

Let's look at the distribution:
```
table(StarWars$agegroup)  # Pretty even...
```

Observed variables:
```
obsvars <- paste0("Q",1:10)
```

Let's look at the mean scores:
```
StarWars %>% group_by(agegroup) %>% summarize_each_(funs(mean), vars = obsvars)
```

Less young people seem to score higher on prequel questions and lower on other questions

Factor loadings matrix:
```
Lambda <- matrix(0, 10, 3)
Lambda[1:4,1] <- 1
Lambda[c(1,5:7),2] <- 1
Lambda[c(1,8:10),3] <- 1
```

Residual covariances:
```
Theta <- diag(1, 10)
Theta[4,10] <- Theta[10,4] <- 1
```

Latents:
```
latents <- c("Prequels","Original","Sequels")
```

Make model:
```
mod_configural <- lvm(StarWars, lambda = Lambda, vars = obsvars, latents = latents, sigma_epsilon = Theta, identification = "variance", groups = "agegroup")
```

Run model:
```
mod_configural <- mod_configural %>% runmodel
```

Look at fit:
```
mod_configurai %>% fit
```

Looks good, let's try weak invariance:
```
mod_weak <- mod_configurai %>% groupequal("lambda") %>% runmodel
```

Compare models:
```
compare(configural = mod_configurai, weak = mod_weak)
```

Weak invariance can be accepted, let's try strong:
```
mod_strong <- mod_weak %>% groupequal("nu") %>% runmodel
```

Means are automatically identified
# Compare models:
choose(configural = mod_configural, weak = mod_weak, strong = mod_strong)

# Questionable p-value and AIC difference, but ok BIC difference. This is quite good, but
# let's take a look. I have not yet implemented LM tests for equality constrains, but we
# can look at something called "equality-free" MIs:
mod_strong %>% MIs(matrices = "nu", type = "free")

# Indicates that Q10 would improve fit. We can also look at residuals:
residuals(mod_strong)

# Let's try freeing intercept 10:
mod_strong_partial <- mod_strong %>% groupfree("nu",10) %>% runmodel

# Compare all models:
choose(configural = mod_configural, weak = mod_weak, strong = mod_strong, strong_partial = mod_strong_partial)

# This seems worth it and lead to an acceptable model! It seems that older people find
# the latest special effects more marvellous!
mod_strong_partial %>% getmatrix("nu")

# Now let's investigate strict invariance:
mod_strict <- mod_strong_partial %>% groupequal("sigma_epsilon") %>% runmodel

# Compare all models:
choose(configural = mod_configural, weak = mod_weak, strong_partial = mod_strong_partial, strict = mod_strict)

# Strict invariance can be accepted!

# Now we can test for homogeneity!
# Are the latent variances equal?
mod_eqvar <- mod_strict %>% groupequal("sigma_zeta") %>% runmodel

# Compare:
choose(strict = mod_strict, eqvar = mod_eqvar)

# This is acceptable. What about the means? (alpha = nu_eta)
mod_eqmeans <- mod_eqvar %>% groupequal("nu_eta") %>% runmodel

# Compare:
choose(eqvar = mod_eqvar, eqmeans = mod_eqmeans)

# Rejected! We could look at MIs again:
mod_eqmeans %>% MIs(matrices = "nu_eta", type = "free")

# Indicates the strongest effect for prequels. Let's see what happens:
eqmeans2 <- mod_eqvar %>%
groupequal("nu_eta",row = c("Original","Sequels")) %>% runmodel
# Compare:
compare(eqvar = mod_eqvar, eqmeans = eqmeans2)

# Questionable, what about the sequels as well?

eqmeans3 <- mod_eqvar %>% groupequal("nu_eta", row = "Original") %>% runmodel

# Compare:
compare(eqvar = mod_eqvar, eqmeans = eqmeans3)

# Still questionable. Let's look at the mean differences:
mod_eqvar %>% getmatrix("nu_eta")

# Looks like people over 30 like the prequels better and the other two trilogies less!

---

**meta_varcov**

**Variance-covariance and GGM meta analysis**

### Description

Meta analysis of correlation matrices to fit a homogenous correlation matrix or Gaussian graphical model. Based on meta-analytic SEM (Jak and Cheung, 2019).

### Usage

```r
```

### Arguments

- **cors**: A list of correlation matrices. Must contain rows and columns with NAs for variables not included in a study.
- **nobs**: A vector with the number of observations per study.
- **Vmats**: Optional list with 'V' matrices (sampling error variance approximations).
**Vmethod**
Which method should be used to approximate the sampling error variance?

**Vestimation**
How should the sampling error estimates be evaluated?

**type**
What to model? Currently only "cor" and "ggm" are supported.

**sigma_y**
Only used when type = "cov". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**kappa_y**
Only used when type = "prec". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**omega_y**
Only used when type = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**lowertri_y**
Only used when type = "chol". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**delta_y**
Only used when type = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**rho_y**
Only used when type = "cor". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**SD_y**
Only used when type = "cor". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**randomEffects**
What to model for the random effects?

**sigma_randomEffects**
Only used when type = "cov". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node
x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**kappa_randomEffects**

Only used when `randomEffects = "prec"`. Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**omega_randomEffects**

Only used when `randomEffects = "gmm"`. Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**lowertri_randomEffects**

Only used when `randomEffects = "chol"`. Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**delta_randomEffects**

Only used when `randomEffects = "gmm"`. Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**rho_randomEffects**

Only used when `randomEffects = "cor"`. Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**SD_randomEffects**

Only used when `randomEffects = "cor"`. Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**vars**

Variables to be included.
baseline_saturated
A logical indicating if the baseline and saturated model should be included. Mostly used internally and NOT Recommended to be used manually.

optimizer
The optimizer to be used. Can be one of "nlminb" (the default R nlminb function), "ucminf" (from the optimr package), and C++ based optimizers "cpp_L-BFGS-B", "cpp_BFGS", "cpp.CG", "cpp_SANN", and "cpp_Nelder-Mead". The C++ optimizers are faster but slightly less stable. Defaults to "nlminb".

estimator
The estimator to be used. Currently implemented are "ML" for maximum likelihood estimation or "FIML" for full-information maximum likelihood estimation.

sampleStats
An optional sample statistics object. Mostly used internally.

verbose
Logical, should progress be printed to the console?

Arguments

x
A psychonetrics model.

all
Logical, should all MIs be printed or only the highest?

matrices
Optional vector of matrices to include in the output.
String indicating which kind of modification index should be printed. ("mi" is the typical MI, "mi_free" is the modification index free from equality constraints across groups, and "mi_equal" is the modification index if the parameter is added constrained equal across all groups).

Number of MIs to include in output if all = FALSE

Logical, should messages be printed?

Logical, should only MIs be printed of non-zero parameters? Useful to explore violations of group equality.

Invisibly returns a relevant subset of the data frame containing all information on the parameters, or a list of such data frames if multiple types of MIs are requested.

Sacha Epskamp

# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")

# Let's take the agreeableness items, and gender:
ConsData <- bfi %>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML"

# Define variables:
vars <- names(ConsData)[1:5]

# Let's fit a full GGM:
mod <- ggm(ConsData, vars = vars, omega = "empty")

# Run model:
mod <- mod %>% runmodel

# Modification indices:
mod %>% MIs
Description

This family is the two-level random intercept variant of the lvm model family. It is mostly a special case of the dlvm family, with the addition of structural effects rather than temporal effects in the beta matrix.

Usage

ml_lnm(...)
ml_rnm(...)
ml_lrnm(...)
ml_lvm(data, lambda, clusters, within_latent = c("cov", "chol", "prec", "ggm"), within_residual = c("cov", "chol", "prec", "ggm"), between_latent = c("cov", "chol", "prec", "ggm"), between_residual = c("cov", "chol", "prec", "ggm"), beta_within = "empty", beta_between = "empty", omega_zeta_within = "full", delta_zeta_within = "full", kappa_zeta_within = "full", sigma_zeta_within = "full", lowertri_zeta_within = "full", omega_epsilon_within = "empty", delta_epsilon_within = "empty", kappa_epsilon_within = "empty", sigma_epsilon_within = "empty", lowertri_epsilon_within = "empty", omega_zeta_between = "full", delta_zeta_between = "full", kappa_zeta_between = "full", sigma_zeta_between = "full", lowertri_zeta_between = "full", omega_epsilon_between = "empty", delta_epsilon_between = "empty", kappa_epsilon_between = "empty", sigma_epsilon_between = "empty", lowertri_epsilon_between = "empty", nu, nu_eta, identify = TRUE, identification = c("loadings", "variance"), vars, latents, groups, equal = "none", baseline_saturated = TRUE, estimator = c("FIML", "MUML"), optimizer, storedata = FALSE, verbose = FALSE, standardize = c("none", "z", "quantile"), sampleStats)

Arguments

data A data frame encoding the data used in the analysis. Must be a raw dataset.
lambda A model matrix encoding the factor loading structure. Each row indicates an indicator and each column a latent. A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains.
For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix. Could also be the result of `simplestructure`.

**clusters**
A string indicating the variable in the dataset that describes group membership.

**within_latent**
The type of within-person latent contemporaneous model to be used.

**within_residual**
The type of within-person residual model to be used.

**between_latent**
The type of between-person latent model to be used.

**between_residual**
The type of between-person residual model to be used.

**beta_within**
A model matrix encoding the within-cluster structural. A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix. Defaults to "empty".

**beta_between**
A model matrix encoding the between-cluster structural. A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix. Defaults to "empty".

**omega_zeta_within**
Only used when `within_latent = "ggm"`. Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**delta_zeta_within**
Only used when `within_latent = "ggm"`. Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**kappa_zeta_within**
Only used when `within_latent = "prec"`. Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**sigma_zeta_within**
Only used when `within_latent = "cov"`. Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**lowertri_zeta_within**
Only used when `within_latent = "chol"`. Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
omega_epsilon_within
Only used when within_residual = "ggm". Can be "full", "empty", or a
typical model matrix with 0s indicating parameters constrained to zero, 1s indi-
cating free parameters, and higher integers indicating equality constrains. For
multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

delta_epsilon_within
Only used when within_residual = "ggm". Can be "full", "empty", or a
typical model matrix with 0s indicating parameters constrained to zero, 1s indi-
cating free parameters, and higher integers indicating equality constrains. For
multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

kappa_epsilon_within
Only used when within_residual = "prec". Can be "full", "empty", or a
typical model matrix with 0s indicating parameters constrained to zero, 1s indi-
cating free parameters, and higher integers indicating equality constrains. For
multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

sigma_epsilon_within
Only used when within_residual = "cov". Can be "full", "empty", or a
typical model matrix with 0s indicating parameters constrained to zero, 1s indi-
cating free parameters, and higher integers indicating equality constrains. For
multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

lowertri_epsilon_within
Only used when within_residual = "chol". Can be "full", "empty", or a
typical model matrix with 0s indicating parameters constrained to zero, 1s indi-
cating free parameters, and higher integers indicating equality constrains. For
multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

omega_zeta_between
Only used when between_latent = "ggm". Can be "full", "empty", or a typi-
cal model matrix with 0s indicating parameters constrained to zero, 1s indicating
free parameters, and higher integers indicating equality constrains. For multiple
groups, this argument can be a list or array with each element/slice encoding such a matrix.

delta_zeta_between
Only used when between_latent = "ggm". Can be "full", "empty", or a typi-
cal model matrix with 0s indicating parameters constrained to zero, 1s indicating
free parameters, and higher integers indicating equality constrains. For multiple
groups, this argument can be a list or array with each element/slice encoding such a matrix.

kappa_zeta_between
Only used when between_latent = "prec". Can be "full", "empty", or a typi-
cal model matrix with 0s indicating parameters constrained to zero, 1s indicating
free parameters, and higher integers indicating equality constrains. For multiple
groups, this argument can be a list or array with each element/slice encoding such a matrix.
sigma_zeta_between
Only used when between_latent = "cov". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

lowertri_zeta_between
Only used when between_latent = "chol". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

omega_epsilon_between
Only used when between_residual = "ggm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

delta_epsilon_between
Only used when between_residual = "ggm". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

kappa_epsilon_between
Only used when between_residual = "prec". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

sigma_epsilon_between
Only used when between_residual = "cov". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

lowertri_epsilon_between
Only used when between_residual = "chol". Can be "full", "empty", or a typical model matrix with 0s indicating parameters constrained to zero, 1s indicating free parameters, and higher integers indicating equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

nu
Optional vector encoding the intercepts of the observed variables. Set elements to 0 to indicate fixed to zero constrains, 1 to indicate free intercepts, and higher integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such a vector.

nu_eta
Optional vector encoding the intercepts of the latent variables. Set elements to 0 to indicate fixed to zero constrains, 1 to indicate free intercepts, and higher
integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such a vector.

**identify**  Logical, should the model be automatically identified?

**identification**  Type of identification used. "loadings" to fix the first factor loadings to 1, and "variance" to fix the diagonal of the latent variable model matrix (\(\sigma_{\zeta},\) lower\(\sigma_{\zeta}\), delta\(\zeta\) or kappa\(\zeta\)) to 1.

**vars**  An optional character vector with names of the variables used.

**latents**  An optional character vector with names of the latent variables.

**groups**  An optional string indicating the name of the group variable in data.

**equal**  A character vector indicating which matrices should be constrained equal across groups.

**baseline_saturated**  A logical indicating if the baseline and saturated model should be included. Mostly used internally and NOT Recommended to be used manually.

**estimator**  Estimator used. Currently only "FIML" is supported.

**optimizer**  The optimizer to be used. Usually either "n1minb" (with box constrains) or "ucminf" (ignoring box constrains), but any optimizer supported by optimr can be used.

**storedata**  Logical, should the raw data be stored? Needed for bootstrapping (see bootstrap).

**verbose**  Logical, should progress be printed to the console?

**standardize**  Which standardization method should be used? "none" (default) for no standardization, "z" for z-scores, and "quantile" for a non-parametric transformation to the quantiles of the marginal standard normal distribution.

**sampleStats**  An optional sample statistics object. Mostly used internally.

...  Arguments sent to `ml_lvm`

**Value**

An object of the class psychonetrics (psychonetrics-class)

**Author(s)**

Sacha Epskamp <mail@sachaepskamp.com>

---

### Description

This function is a wrapper around dlv1m that allows for specifying the model using a long format data and similar input as the mlVAR package. The ml_tsd1vm simply sets `within_latent = "ggm"` and `between_latent = "ggm"` by default. The ml_gvar and ml_var are simple wrappers with different named defaults for contemporaneous and between-person effects.
Usage

```r
ml_tslvm1(data, beepvar, idvar, vars, groups, estimator = "FIML",
          standardize = c("none", "z", "quantile"), ...)
ml_ts_lvgvar(...)
ml_gvar(..., contemporaneous = c("ggm", "cov", "chol", "prec"),
          between = c("ggm", "cov", "chol", "prec"))
ml_var(..., contemporaneous = c("cov", "chol", "prec", "ggm"),
          between = c("cov", "chol", "prec", "ggm"))
```

Arguments

- **data**: The data to be used. Must be raw data in long format (each row indicates one person at one time point).
- **beepvar**: Optional string indicating assessment beep per day. Adding this argument will cause non-consecutive beeps to be treated as missing!
- **idvar**: String indicating the subject ID
- **vars**: Vectors of variables to include in the analysis
- **groups**: An optional string indicating the name of the group variable in data.
- **estimator**: Estimator to be used. Must be "FIML".
- **standardize**: Which standardization method should be used? "none" (default) for no standardization, "z" for z-scores, and "quantile" for a non-parametric transformation to the quantiles of the marginal standard normal distribution.
- **contemporaneous**: The type of within-person latent contemporaneous model to be used.
- **between**: The type of between-person latent model to be used.
- **...**: Arguments sent to `dlvm1`

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

---

**modelsearch**

*Stepwise model search*

Description

This function performs stepwise model search to find an optimal model that (locally) minimizes some criterion (by default, the BIC).
Usage

```
modelsearch(x, criterion = "bic", matrices, prunealpha = 0.01,
            addalpha = 0.01, verbose, ...)
```

Arguments

- **x**: A psychonetrics model.
- **criterion**: String indicating the criterion to minimize. Any criterion from `fit` can be used.
- **matrices**: Vector of strings indicating which matrices should be searched. Will default to network structures and factor loadings.
- **prunealpha**: Minimal alpha used to consider edges to be removed
- **addalpha**: Maximum alpha used to consider edges to be added
- **verbose**: Logical, should messages be printed?
- **...**: Arguments sent to `runmodel`

Details

The full algorithm is as follows:
1. Evaluate all models in which an edge is removed that has $p > \text{prunealpha}$, or an edge is added that has a modification index with $p < \text{addalpha}$
2. If none of these models improve the criterion, return the previous model and stop the algorithm
3. Update the model to the model that improved the criterion the most
4. Evaluate all other considered models that improved the criterion
5. If none of these models improve the criterion, go to 1, else go to 3

Value

An object of the class psychonetrics (`psychonetrics-class`)

Author(s)

Sacha Epskamp

See Also

`prune`, `stepup`

Examples

```
# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")
```
# Let's take the agreeableness items, and gender:
ConsData <- bfi %>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML)

# Define variables:
vars <- names(ConsData)[1:5]

# Let's fit a full GGM:
mod <- ggm(ConsData, vars = vars)

# Run model:
mod <- mod %>% runmodel

# Model search
mod <- mod %>% prune(alpha = 0.01) %>% modelsearch

---

### parameters

**Description**

This function will print a list of parameters of the model

**Usage**

`parameters(x)`

**Arguments**

- `x` A psychonetrics model.

**Value**

Invisibly returns a data frame containing information on all parameters.

**Author(s)**

Sacha Epskamp

**Examples**

# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")
# Let's take the agreeableness items, and gender:
ConsData <- bfi %>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML)

# Define variables:
vars <- names(ConsData)[1:5]

# Let's fit a full GGM:
mod <- ggm(ConsData, vars = vars, omega = "empty")

# Run model:
mod <- mod %>% runmodel

# Parameter estimates:
mod %>% parameters

---

**parequal**

*Set equality constrains across parameters*

**Description**

This function can be used to set parameters equal

**Usage**

```r
parequal(x, ..., inds = integer(0), verbose, log = TRUE, runmodel = FALSE)
```

**Arguments**

- `x`: A psychonetrics model.
- `...`: Arguments sent to runmodel
- `inds`: Parameter indices of parameters to be constrained equal
- `verbose`: Logical, should messages be printed?
- `log`: Logical, should the log be updated?
- `runmodel`: Logical, should the model be updated?

**Value**

An object of the class psychonetrics (*psychonetrics-class*)

**Author(s)**

Sacha Epskamp
**Description**

This function will search for a multi-group model with equality constrains on some but not all parameters. This is called partial pruning (Epskamp, Isvoranu, & Cheung, 2020; Haslbeck, 2020). The algorithm is as follows: 1. remove all parameters not significant at alpha in all groups (without equality constrains), 2. create a union model with all parameters included in any group included in all groups and constrained equal. 3. Stepwise free equality constrains of the parameter that features the largest sum of modification indices until BIC can no longer be improved. 4. Select and return the best model according to BIC (original model, pruned model, union model and partially pruned model).

**Usage**

`partialprune(x, alpha = 0.01, matrices, verbose, combinefun = unionmodel, ...)`

**Arguments**

- `x` A psychonetrics model.
- `alpha` Significance level to use.
- `matrices` Vector of strings indicating which matrices should be pruned. Will default to network structures.
- `verbose` Logical, should messages be printed?
- `combinefun` Function used to combine models of different groups.
- `...` Arguments sent to `prune`.

**Author(s)**

Sacha Epskamp <mail@sachaepskamp.com>

**References**


prune

Stepdown model search by pruning non-significant parameters.

Description

This function will (recursively) remove parameters that are not significant and refit the model.

Usage

prune(x, alpha = 0.01, adjust = c("none", "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr"), matrices, runmodel = TRUE, recursive = FALSE, verbose, log = TRUE, identify = TRUE, startreduce = 1, limit = Inf, mode = c("tested","all"), ...)

Arguments

x A psychonetrics model.
alpha Significance level to use.
adjust p-value adjustment method to use. See p.adjust.
matrices Vector of strings indicating which matrices should be pruned. Will default to network structures.
runmodel Logical, should the model be evaluated after pruning?
recursive Logical, should the pruning process be repeated?
verbose Logical, should messages be printed?
log Logical, should the log be updated?
identify Logical, should models be identified automatically?
startreduce A numeric value indicating a factor with which the starting values should be reduced. Can be useful when encountering numeric problems.
limit The maximum number of parameters to be pruned.
mode Mode for adjusting for multiple comparisons. Should all parameters be considered as the total number of tests or only the tested parameters (parameters of interest)?
... Arguments sent to runmodel

Value

An object of the class psychonetrics (psychonetrics-class)

Author(s)

Sacha Epskamp
See Also

stepup

Examples

# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")

# Let's take the agreeableness items, and gender:
ConsData <- bfi %>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML)

# Define variables:
vars <- names(ConsData)[1:5]

# Let's fit a full GGM:
mod <- ggm(ConsData, vars = vars, omega = "full")

# Run model:
mod <- mod %>% runmodel

# Prune model:
mod <- mod %>% prune(adjust = "fdr", recursive = FALSE)

psychonetrics-class

Class "psychonetrics"

Description

Main class for psychonetrics results.

Objects from the Class

Objects can be created by calls of the form new("psychonetrics", ...).

Slots

model: Object of class "character"
submodel: Object of class "character"
parameters: Object of class "data.frame"
matrices: Object of class "data.frame"
meanstructure: Object of class "logical"
computed: Object of class "logical" ~~
sample: Object of class "psychonetrics_samplestats" ~~
modelmatrices: Object of class "list" ~~
log: Object of class "psychonetrics_log" ~~
optim: Object of class "list" ~~
fitmeasures: Object of class "list" ~~
baseline_saturated: Object of class "list" ~~
equal: Object of class "character" ~~
objective: Object of class "numeric" ~~
information: Object of class "matrix" ~~
identification: Object of class "character" ~~
optimizer: Object of class "character" ~~
optim.args: Object of class "list" ~~
estimator: Object of class "character" ~~
distribution: Object of class "character" ~~
extramatrices: Object of class "list" ~~
rawts: Object of class "logical" ~~
Drawts: Object of class "list" ~~
types: Object of class "list" ~~
cpp: Object of class "logical" ~~
verbose: Object of class "logical" ~~

Methods

resid signature(object = "psychonetrics"): ...
residuals signature(object = "psychonetrics"): ...
show signature(object = "psychonetrics"): ...

Author(s)

Sacha Epskamp

Examples

showClass("psychonetrics")
Model updating functions

Description
These functions update a psychonetrics model. Typically they are not required.

Usage

\[
\text{addMIs}(x, \text{matrices} = \text{"all"}, \text{type} = \text{c("normal", "free", "equal"), verbose, analyticFisher = TRUE})
\]

\text{addSEs}(x, \text{verbose})

\text{addfit}(x, \text{verbose})

\text{identify}(x)

Arguments

- **x**: A psychonetrics model.
- **matrices**: Optional vector of matrices to include in MIs.
- **type**: String indicating which modification indices should be updated.
- **verbose**: Logical, should messages be printed?
- **analyticFisher**: Logical indicating if an analytic Fisher information matrix should be used.

Value

An object of the class psychonetrics (psychonetrics-class)

Author(s)

Sacha Epskamp
runmodel

Run a psychonetrics model

Description

This is the main function used to run a psychonetrics model.

Usage

runmodel(x, level = c("gradient", "fitfunction"), addfit = TRUE, addMIs = TRUE, addSEs = TRUE, addInformation = TRUE, log = TRUE, verbose, optim.control, analyticFisher = TRUE, warn_improper = TRUE, warn_gradient = TRUE, return_improper = TRUE, bounded = TRUE)

Arguments

x A psychonetrics model.
level Level at which the model should be estimated. Defaults to "gradient" to indicate the analytic gradient should be used.
addfit Logical, should fit measures be added?
addMIs Logical, should modification indices be added?
addSEs Logical, should standard errors be added?
addInformation Logical, should the Fisher information be added?
log Logical, should the log be updated?
verbose Logical, should messages be printed?
optim.control A list with options for optimr
analyticFisher Logical, should the analytic Fisher information be used? If FALSE, numeric information is used instead.
return_improper Should a result in which improper computation was used be return? Improper computation can mean that a pseudoinverse of small spectral shift was used in computing the inverse of a matrix.
warn_improper Logical. Should a warning be given when at some point in the estimation a pseudoinverse was used?
warn_gradient Logical. Should a warning be given when the average absolute gradient is > 1?
bounded Logical. Should bounded estimation be used (e.g., variances should be positive)?

Value

An object of the class psychonetrics (psychonetrics-class)
Author(s)
Sacha Epskamp

Examples

# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")

# Let's take the agreeableness items, and gender:
ConsData <- bfi %>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML)

# Define variables:
vars <- names(ConsData)[1:5]

# Let's fit a full GGM:
mod <- ggm(ConsData, vars = vars, omega = "full")

# Run model:
mod <- mod %>% runmodel

setestimator
Convenience functions

Description
These functions can be used to change some estimator options.

Usage

setestimator(x, estimator)

setoptimizer(x, optimizer = c("default", "nlminb", "ucminf",
  "cpp_L-BFGS-B", "cpp_BFGS", "cpp_CG", "cpp_SANN",
  "cpp_Nelder-Mead"), optim.args)

usecpp(x, use = TRUE)

Arguments

x A psychonetrics model.
estimator A string indicating the estimator to be used
optimizer

The optimizer to be used. Can be one of "nlminb" (the default R `nlminb` function), "ucminf" (from the optimr package), and C++ based optimizers "cpp_L-BFGS-B", "cpp_BFGS", "cpp_CG", "cpp_SANN", and "cpp_Nelder-Mead". The C++ optimizers are faster but slightly less stable. Defaults to "nlminb".

use

Logical indicating if C++ should be used (currently only used in FIML)

optim.args

List of arguments to sent to the optimizer.

Details

The default optimizer is nlminb with the following arguments:

- eval.max=20000L
- iter.max=10000L
- trace=0L
- abs.tol=sqrt(.Machine$double.eps)
- rel.tol=sqrt(.Machine$double.eps)
- step.min=1.0
- step.max=1.0
- x.tol=1.5e-8
- xf.tol=2.2e-14

Value

An object of the class psychonetrics (psychonetrics-class)

Author(s)

Sacha Epskamp

Description

This function controls if messages should be printed for a psychonetrics model.

Usage

`setverbose(x, verbose = TRUE)`

Arguments

- `x` A psychonetrics model.
- `verbose` Logical indicating if verbose should be enabled
Value

An object of the class psychonetrics (psychonetrics-class)

Author(s)

Sacha Epskamp

---

**simplestructure**  Generate factor loadings matrix with simple structure

---

Description

This function generates the input for lambda arguments in latent variable models using a simple structure. The input is a vector with an element for each variable indicating the factor the variable loads on.

Usage

`simplestructure(x)`

Arguments

- `x` A vector indicating which factor each indicator loads on.

Author(s)

Sacha Epskamp <mail@sachaepskamp.com>

---

**StarWars**  Star Wars dataset

---

Description

This questionnaire was constructed by Carolin Katzer, Charlotte Tanis, Esther Niehoff, Myrthe Veenman, and Jason Nak as part of an assignment for a course on confirmatory factor analysis (http://sachaepskamp.com/SEM2018). They also collected the data among fellow psychology students as well as through social media.

Usage

`data("StarWars")`
Format

A data frame with 271 observations on the following 13 variables.

Q1 I am a huge Star Wars fan! (star what?)
Q2 I would trust this person with my democracy <picture of Jar Jar Binks>
Q3 I enjoyed the story of Anakin’s early life
Q4 The special effects in this scene are awful <video of the Battle of Geonosis>
Q5 I would trust this person with my life <picture of Han Solo>
Q6 I found Darth Vader’s big reveal in "Empire" one of the greatest moments in movie history
Q7 The special effects in this scene are amazing <video of the Death Star explosion>
Q8 If possible, I would definitely buy this droid <picture of BB-8>
Q9 The story in the Star Wars sequels is an improvement to the previous movies
Q10 The special effects in this scene are marvellous <video of the Starkiller Base firing>
Q11 What is your gender?
Q12 How old are you?
Q13 Have you seen any of the Star Wars movies?

Details

The questionnaire is online at https://github.com/SachaEpskamp/SEM-code-examples/blob/master/CFA_fit_examples/StarWars_questionaire.pdf. The authors of the questionnaire defined a measurement model before collecting data: Q2 - Q4 are expected to load on a "prequel" factor, Q5 - Q7 are expected to load in a "originals" factor, and Q8 - Q10 are expected to load on a "sequal" factor. Finally, Q1 is expected to load on all three.

Source

https://github.com/SachaEpskamp/SEM-code-examples/blob/master/CFA_fit_examples

Examples

data(StarWars)

stepup

Stepup model search along modification indices

Description

This function automatically performs step-up search by adding the parameter with the largest modification index until some criterion is reached or no modification indices are significant at alpha.
Usage

stepup(x, alpha = 0.01, criterion = "bic", matrices, mi = c("mi", "mi_free", "mi_equal")), greedyadjust = c("bonferroni", "none", "holm", "hochberg", "hommel", "fdr", "BH", "BY"), stopif, greedy = FALSE, verbose, checkinformation = TRUE, singularinformation = c("tryfix", "skip", "continue", "stop"), startEPC = TRUE, ...)  

Arguments

x  A psychonetrics model.
alpha  Significance level to use.
criterion  String indicating the criterion to minimize. Any criterion from fit can be used.
matrices  Vector of strings indicating which matrices should be searched. Will default to network structures and factor loadings.
mi  String indicating which kind of modification index should be used ("mi" is the typical MI, "mi_free" is the modification index free from equality constrains across groups, and "mi_equal" is the modification index if the parameter is added constrained equal across all groups).
greedyadjust  String indicating which p-value adjustment should be used in greedy start. Any method from p.adjust can be used.
stopif  An R expression, using objects from fit, which will break stepup search if it evaluates to TRUE. For example, stopif = rmsea < 0.05 will lead to search to stop if rmsea is below 0.05.
greedy  Logical, should a greedy start be used? If TRUE, the first step adds any parameter that is significant (after adjustment)
verbose  Logical, should messages be printed?
checkinformation  Logical, should the Fisher information be checked for potentially non-identified models?
singularinformation  String indicating how to proceed if the information matrix is singular. "tryfix" will adjust starting values to try to fix the problem, "skip" will lead to the algorithm to skip the current parameter, "continue" will ignore the situation, and "stop" will break the algorithm and return a list with the last two models.
startEPC  Logical, should the starting value be set at the expected parameter change?
...  Arguments sent to runmodel

Value

An object of the class psychonetrics (psychonetrics-class)

Author(s)

Sacha Epskamp
See Also

prune

Examples

```r
# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")

# Let's take the agreeableness items, and gender:
ConsData <- bfi |>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML"

# Define variables:
vars <- names(ConsData)[1:5]

# Let's fit a full GGM:
mod <- ggm(ConsData, vars = vars, omega = "full")

# Run model:
mod <- mod %>%runmodel %>%prune(alpha = 0.05)

# Remove an edge (example):
mod <- mod %>%fixpar("omega",1,2) %>%runmodel

# Stepup search
mod <- mod %>%stepup(alpha = 0.05)
```

---

tsdLVm1

Lag-1 dynamic latent variable model family of psychonetrics models for time-series data

Description

This is the family of models that models a dynamic factor model on time-series. There are two covariance structures that can be modeled in different ways: contemporaneous for the contemporaneous model and residual for the residual model. These can be set to "cov" for covariances, "prec" for a precision matrix, "ggm" for a Gaussian graphical model and "chol" for a Cholesky decomposition. The ts.lvGvar wrapper function sets contemporaneous = "ggm" for the graphical VAR model.
tsdlvm1\(\text{df, lambda, contemporaneous = c("cov", "chol", "prec", "gmm", residual = c("cov", "chol", "prec", "gmm", beta = "full", omega_zeta = "full", delta_zeta = "full", kappa_zeta = "full", sigma_zeta = "full", lowertri_zeta = "full", omega_epsilon = "empty", delta_epsilon = "empty", kappa_epsilon = "empty", sigma_epsilon = "empty", lowertri_epsilon = "empty", nu, mu_eta, identify = TRUE, identification = c("loadings", "variance"), latents, beepvar, dayvar, idvar, vars, groups, covs, means, nobs, missing = "listwise", equal = "none", baseline_saturated = TRUE, estimator = "ML", optimizer, storedata = FALSE, sampleStats, covtype = c("choose", "ML", "UB"), centerWithin = FALSE, standardize = c("none", "z", "quantile"), verbose = FALSE)}

\text{ts_lvgvar(...)}

**Arguments**

- **data**
  A data frame encoding the data used in the analysis. Can be missing if \text{covs} and \text{nobs} are supplied.

- **lambda**
  A model matrix encoding the factor loading structure. Each row indicates an indicator and each column a latent. A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

- **contemporaneous**
  The type of contemporaneous model used. See description.

- **residual**
  The type of residual model used. See description.

- **beta**
  A model matrix encoding the temporal relationships (transpose of temporal network) between latent variables. A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix. Can also be "full" for a full temporal network or "empty" for an empty temporal network.

- **omega_zeta**
  Only used when \text{contemporaneous = "gmm"}. Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

- **delta_zeta**
  Only used when \text{contemporaneous = "gmm"}. Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to
estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**kappa_zeta** Only used when `contemporaneous = "prec"`. Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**sigma_zeta** Only used when `contemporaneous = "cov"`. Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**lowertri_zeta** Only used when `contemporaneous = "chol"`. Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**omega_epsilon** Only used when `residual = "ggm"`. Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**delta_epsilon** Only used when `residual = "ggm"`. Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**kappa_epsilon** Only used when `residual = "prec"`. Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**sigma_epsilon** Only used when `residual = "cov"`. Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

**lowertri_epsilon** Only used when `residual = "chol"`. Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
freely, "empty" to only include diagonal elements, or a matrix of the dimensions
node x node with 0 encoding a fixed to zero element, 1 encoding a free to es-
timate element, and higher integers encoding equality constrains. For multiple
groups, this argument can be a list or array with each element/slice encoding such a matrix.

nu Optional vector encoding the intercepts of the observed variables. Set elements
to 0 to indicate fixed to zero constrains, 1 to indicate free intercepts, and higher
integers to indicate equality constrains. For multiple groups, this argument can
be a list or array with each element/column encoding such a vector.

mu_eta Optional vector encoding the means of the latent variables. Set elements to 0 to
indicate fixed to zero constrains, 1 to indicate free intercepts, and higher integers
to indicate equality constrains. For multiple groups, this argument can be a list
or array with each element/column encoding such a vector.

identify Logical, should the model be automatically identified?

identification Type of identification used. "loadings" to fix the first factor loadings to 1, and
"variance" to fix the diagonal of the latent variable model matrix (sigma_zeta,
lowertri_zeta, delta_zeta or kappa_zeta) to 1.

latents An optional character vector with names of the latent variables.

beepvar Optional string indicating assessment beep per day. Adding this argument will
cause non-consecutive beeps to be treated as missing!

dayvar Optional string indicating assessment day. Adding this argument makes sure that
the first measurement of a day is not regressed on the last measurement of the
previous day. IMPORTANT: only add this if the data has multiple observations
day.

idvar Optional string indicating the subject ID

vars An optional character vector encoding the variables used in the analysis. Must
equal names of the dataset in data.

groups An optional string indicating the name of the group variable in data.

covs A sample variance–covariance matrix, or a list/array of such matrices for multi-
ple groups. Make sure covtype argument is set correctly to the type of covari-
ances used.

means A vector of sample means, or a list/matrix containing such vectors for multiple
groups.

nobs The number of observations used in covs and means, or a vector of such num-
bers of observations for multiple groups.

missing How should missingness be handled in computing the sample covariances and
number of observations when data is used. Can be "listwise" for listwise
deletion, or "pairwise" for pairwise deletion.

equal A character vector indicating which matrices should be constrained equal across
groups.

baseline_saturated A logical indicating if the baseline and saturated model should be included.
Mostly used internally and NOT Recommended to be used manually.
estimator  The estimator to be used. Currently implemented are "ML" for maximum likelihood estimation, "FIML" for full-information maximum likelihood estimation, "ULS" for unweighted least squares estimation, "WLS" for weighted least squares estimation, and "DWLS" for diagonally weighted least squares estimation.

optimizer  The optimizer to be used. Can be one of "nlminb" (the default R nlminb function), "ucminf" (from the optimr package), and C++ based optimizers "cpp_L-BFGS-B", "cpp_BFGS", "cpp.CG", "cpp_SANN", and "cpp_Nelder-Mead". The C++ optimizers are faster but slightly less stable. Defaults to "nlminb".

storedata  Logical, should the raw data be stored? Needed for bootstrapping (see bootstrap).

standardize  Which standardization method should be used? "none" (default) for no standardization, "z" for z-scores, and "quantile" for a non-parametric transformation to the quantiles of the marginal standard normal distribution.

sampleStats  An optional sample statistics object. Mostly used internally.

centerWithin  Logical, should data be within-person centered?

covtype  If `cova` is used, this is the type of covariance (maximum likelihood or unbiased) the input covariance matrix represents. Set to "ML" for maximum likelihood estimates (denominator n) and "UB" to unbiased estimates (denominator n-1). The default will try to find the type used, by investigating which is most likely to result from integer valued datasets.

verbose  Logical, should messages be printed?

Value
An object of the class psychonetrics (psychonetrics-class)

Author(s)
Sacha Epskamp

Examples

# Note: this example is wrapped in a dontrun environment because the data is not
# available locally.
## Not run:
# Obtain the data from:
# Epskamp, S., van Borkulo, C. D., van der Veen, D. C., Servaas, M. N., Isvoranu, A. M.,
# The importance of contemporaneous and temporal connections. Clinical Psychological
# Science, 6(3), 416-427.
# Available here: https://osf.io/c8wjz/
# Encode time variable in a way R understands:
# tdata <- as.POSIXct(tdata$time, tz = "Europe/Amsterdam")

# Encode time variable in a way R understands:
tdata$time <- as.POSIXct(tdata$time, tz = "Europe/Amsterdam")
# Extract days:
tsdata$Day <- as.Date(tsdata$time, tz = "Europe/Amsterdam")

# Variables to use:
vars <- c("relaxed", "sad", "nervous", "concentration", "tired", "rumination",
           "bodily.discomfort")

# Create lambda matrix (in this case: one factor):
Lambda <- matrix(1,7,1)

# Estimate dynamical factor model:
model <- tsdlvm1(
  tsdata,
  lambda = Lambda,
  vars = vars,
  dayvar = "Day",
  estimator = "FIML"
)

# Run model:
model <- model %>% runmodel

# Look at fit:
model %>% print
model %>% fit # Pretty bad fit

## End(Not run)

---

unionmodel  

**Unify models across groups**

**Description**

The unionmodel will add all parameters to all groups that are free in at least one group, and the intersectionmodel will constrain all parameters across groups to zero unless they are free to estimate in all groups.

**Usage**

```r
unionmodel(x, runmodel = FALSE, verbose, log = TRUE, identify = TRUE, ...)
```

```r
intersectionmodel(x, runmodel = FALSE, verbose, log = TRUE, identify = TRUE, ...)
```

**Arguments**

- **x**  
  A psychonetrics model.
runmodel Logical, should the model be updated?
verbose Logical, should messages be printed?
log Logical, should the log be updated?
identify Logical, should the model be identified?
... Arguments sent to runmodel

Value
An object of the class psychonetrics (psychonetrics-class)

Author(s)
Sacha Epskamp

---

Description
This is the family of models that models time-series data using a lag-1 vector autoregressive model (VAR: Epskamp, Waldorp, Mottus, Borsboom, 2018). The model is fitted to the Toeplitz matrix, but unlike typical SEM software the block of covariances of the lagged variables is not used in estimating the temporal and contemporaneous relationships (the block is modeled completely separately using a cholesky decomposition, and does not enter the model elsewise). The contemporaneous argument can be used to define what contemporaneous model is used: contemporaneous = "cov" (default) models a variance-covariance matrix, contemporaneous = "chol" models a Cholesky decomposition, contemporaneous = "prec" models a precision matrix, and contemporaneous = "ggm" (alias: gvar()) models a Gaussian graphical model, also then known as a graphical VAR model.

Usage

```r
var1(data, contemporaneous = c("cov", "chol", "prec", "ggm"), beta = "full", omega_zeta = "full", delta_zeta = "full", kappa_zeta = "full", sigma_zeta = "full", lowertri_zeta = "full", mu, beepvar, dayvar, idvar, vars, groups, covs, means, nobs, missing = "listwise", equal = "none", baseline_saturated = TRUE, estimator = "ML", optimizer, storedata = FALSE, covtype = c("choose", "ML", "UB"), standardize = c("none", "z", "quantile"), sampleStats, verbose = FALSE, bootstrap = FALSE)
```

gvar(...)
Arguments

data  A data frame encoding the data used in the analysis. Can be missing if covs and nobs are supplied.

contemporaneous  The type of contemporaneous model used. See description.

beta  A model matrix encoding the temporal relationships (transpose of temporal network). A 0 encodes a fixed to zero element, a 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix. Can also be "full" for a full temporal network or "empty" for an empty temporal network.

omega_zeta  Only used when contemporaneous = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

delta_zeta  Only used when contemporaneous = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

kappa_zeta  Only used when contemporaneous = "prec". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

sigma_zeta  Only used when contemporaneous = "cov". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

lowertri_zeta  Only used when contemporaneous = "chol". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.

mu  Optional vector encoding the mean structure. Set elements to 0 to indicate fixed to zero constrains, 1 to indicate free means, and higher integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such a vector.
Optional string indicating assessment beep per day. Adding this argument will cause non-consecutive beeps to be treated as missing!

Optional string indicating assessment day. Adding this argument makes sure that the first measurement of a day is not regressed on the last measurement of the previous day. IMPORTANT: only add this if the data has multiple observations per day.

Optional string indicating the subject ID

An optional character vector encoding the variables used in the analysis. Must equal names of the dataset in data.

An optional string indicating the name of the group variable in data.

A sample variance–covariance matrix, or a list/array of such matrices for multiple groups. Make sure covtype argument is set correctly to the type of covariances used.

A vector of sample means, or a list/matrix containing such vectors for multiple groups.

The number of observations used in covs and means, or a vector of such numbers of observations for multiple groups.

How should missingness be handled in computing the sample covariances and number of observations when data is used. Can be "listwise" for listwise deletion, or "pairwise" for pairwise deletion.

A character vector indicating which matrices should be constrained equal across groups.

A logical indicating if the baseline and saturated model should be included. Mostly used internally and NOT Recommended to be used manually.

The estimator to be used. Currently implemented are "ML" for maximum likelihood estimation, "FIML" for full-information maximum likelihood estimation, "ULS" for unweighted least squares estimation, "WLS" for weighted least squares estimation, and "DWLS" for diagonally weighted least squares estimation.

The optimizer to be used. Can be one of "nlminb" (the default R nlminb function), "ucminf" (from the optimr package), and C++ based optimizers "cpp_L-BFGS-B", "cpp_BFGS", "cpp_CG", "cpp_SANN", and "cpp_Nelder-Mead". The C++ optimizers are faster but slightly less stable. Defaults to "nlminb".

Logical, should the raw data be stored? Needed for bootstrapping (see bootstrap).

Which standardization method should be used? "none" (default) for no standardization, "z" for z-scores, and "quantile" for a non-parametric transformation to the quantiles of the marginal standard normal distribution.

An optional sample statistics object. Mostly used internally.

If `covs` is used, this is the type of covariance (maximum likelihood or unbiased) the input covariance matrix represents. Set to "ML" for maximum likelihood estimates (denominator n) and "UB" to unbiased estimates (denominator n-1). The default will try to find the type used, by investigating which is most likely to result from integer valued datasets.
verbose Logical, should messages be printed?
bootstrap Bootstraps the data (reshuffles rows with replacement).
... Arguments sent to var1

Details
This will be updated in a later version.

Value
An object of the class psychonetrics

Author(s)
Sacha Epskamp

References

See Also
lvm, varcov, dlvm

Examples
library("dplyr")
library("graphicalVAR")

beta <- matrix(c( 0, 0.5, 0.5, 0), 2, 2, byrow=TRUE)
kappa <- diag(2)
simData <- graphicalVARsim(50, beta, kappa)

# Form model:
model <- gvar(simData)

# Evaluate model:
model <- model %>% runmodel

# Parameter estimates:
model %>% parameters

# Plot the CIs:
CIplot(model, "beta")

# Note: this example is wrapped in a dontrun environment because the data is not available locally.
## Not run:
# Longer example:
#
# Obtain the data from:
#
# Epskamp, S., van Borkulo, C. D., van der Veen, D. C., Servaas, M. N., Isvoranu, A. M.,
# The importance of contemporaneous and temporal connections. Clinical Psychological
# Science, 6(3), 416-427.
#
# Available here: https://osf.io/c8wjz/

tsdata <- read.csv("Supplementary2_data.csv")

# Encode time variable in a way R understands:
tsdata$time <- as.POSIXct(tsdata$time, tz = "Europe/Amsterdam")

# Extract days:
tsdata$Day <- as.Date(tsdata$time, tz = "Europe/Amsterdam")

# Variables to use:
vars <- c("relaxed", "sad", "nervous", "concentration", "tired", "rumination",
"bodily.discomfort")

# Estimate, prune with FDR, and perform stepup search:
model_FDRprune <- gvar(
tsdata,
vars = vars,
dayvar = "Day",
estimator = "FIML"
) %>%
runmodel %>%
prune(adjust = "fdr", recursive = FALSE) %>%
stepup(criterion = "bic")

# Estimate with greedy stepup search:
model_stepup <- gvar(
tsdata,
vars = vars,
dayvar = "Day",
estimator = "FIML",
omega_zeta = "empty",
beta = "empty"
) %>%
runmodel %>%
stepup(greedy = TRUE, greedyadjust = "bonferroni", criterion = "bic")

# Compare models:
compare(
FDRprune = model_FDRprune,
stepup = model_stepup
)

# Very similar but not identical. Stepup is preferred here according to AIC and BIC
# Stepup results:
temporal <- getmatrix(model_stepup, "PDC") # PDC = Partial Directed Correlations
contemporaneous <- getmatrix(model_stepup, "omega_zeta")

# Average layout:
library("qgraph")
L <- averageLayout(temporal, contemporaneous)

# Labels:
labs <- gsub("\"","\n",vars)

# Plot:
layout(t(1:2))
qgraph(temporal, layout = L, theme = "colorblind", directed=TRUE, diag=TRUE,
  title = "Temporal", vsize = 12, mar = rep(6,4), asize = 5,
  labels = labs)
qgraph(contemporaneous, layout = L, theme = "colorblind",
  title = "Contemporaneous", vsize = 12, mar = rep(6,4), asize = 5,
  labels = labs)
## End(Not run)

## varcov

Variance-covariance family of psychonetrics models

### Description

This is the family of models that models only a variance-covariance matrix with mean structure. The type argument can be used to define what model is used: type = "cov" (default) models a variance-covariance matrix directly, type = "chol" (alias: cholesky()) models a Cholesky decomposition, type = "prec" (alias: precision()) models a precision matrix, type = "ggm" (alias: ggm()) models a Gaussian graphical model (Epskamp, Rhemtulla and Borsboom, 2017), and type = "cor" (alias: corr()) models a correlation matrix.

### Usage

```
varcov(data, type = c("cov", "chol", "prec", "ggm", "cor"),
  sigma = "full", kappa = "full", omega = "full",
  lowertri = "full", delta = "full", rho = "full", SD =
  "full", mu, tau, vars, ordered = character(0), groups,
  covs, means, nobs, missing = "listwise", equal =
  "none", baseline_saturated = TRUE, estimator =
  "default", optimizer, storedata = FALSE, WLS.W,
  sampleStats, meanstructure, corinput, verbose = FALSE,
  covtype = c("choose", "ML", "UB"), standardize =
  c("none", "z", "quantile"), fullFIML = FALSE)
```

`cholesky(...)`

`precision(...)`

Arguments

data A data frame encoding the data used in the analysis. Can be missing if covs and nobs are supplied.
type The type of model used. See description.
sigma Only used when type = "cov". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
kappa Only used when type = "prec". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
omega Only used when type = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
lowertri Only used when type = "chol". Either "full" to estimate every element freely, "empty" to only include diagonal elements, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
delta Only used when type = "ggm". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
rho Only used when type = "cor". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
SD Only used when type = "cor". Either "full" to estimate every element freely, "empty" to set all elements to zero, or a matrix of the dimensions node x node with 0 encoding a fixed to zero element, 1 encoding a free to estimate element, and higher integers encoding equality constrains. For multiple groups, this argument can be a list or array with each element/slice encoding such a matrix.
mu

Optional vector encoding the mean structure. Set elements to 0 to indicate fixed to zero constrains, 1 to indicate free means, and higher integers to indicate equality constrains. For multiple groups, this argument can be a list or array with each element/column encoding such a vector.

tau

Optional list encoding the thresholds per variable.

vars

An optional character vector encoding the variables used in the analysis. Must equal names of the dataset in data.

groups

An optional string indicating the name of the group variable in data.

covs

A sample variance–covariance matrix, or a list/array of such matrices for multiple groups. Make sure covtype argument is set correctly to the type of covariances used.

means

A vector of sample means, or a list/matrix containing such vectors for multiple groups.

nobs

The number of observations used in covs and means, or a vector of such numbers of observations for multiple groups.

covtype

If 'covs' is used, this is the type of covariance (maximum likelihood or unbiased) the input covariance matrix represents. Set to "ML" for maximum likelihood estimates (denominator n) and "UB" to unbiased estimates (denominator n-1). The default will try to find the type used, by investigating which is most likely to result from integer valued datasets.

missing

How should missingness be handled in computing the sample covariances and number of observations when data is used. Can be "listwise" for listwise deletion, or "pairwise" for pairwise deletion.

equal

A character vector indicating which matrices should be constrained equal across groups.

baseline_saturated

A logical indicating if the baseline and saturated model should be included. Mostly used internally and NOT Recommended to be used manually.

estimator

The estimator to be used. Currently implemented are "ML" for maximum likelihood estimation, "FIML" for full-information maximum likelihood estimation, "ULS" for unweighted least squares estimation, "WLS" for weighted least squares estimation, and "DWLS" for diagonally weighted least squares estimation.

optimizer

The optimizer to be used. Can be one of "nlminb" (the default R nlminb function), "ucminf" (from the optimr package), and C++ based optimizers "cpp_L-BFGS-B", "cpp_BFGS", "cpp_CG", "cpp_SANN", and "cpp_Nelder-Mead". The C++ optimizers are faster but slightly less stable. Defaults to "nlminb".

storedata

Logical, should the raw data be stored? Needed for bootstrapping (see bootstrap).

standardize

Which standardization method should be used? "none" (default) for no standardization, "z" for z-scores, and "quantile" for a non-parametric transformation to the quantiles of the marginal standard normal distribution.

WLS.W

Optional WLS weights matrix.

sampleStats

An optional sample statistics object. Mostly used internally.

verbose

Logical, should progress be printed to the console?
ordered A vector with strings indicating the variables that are ordered categorical, or set to TRUE to model all variables as ordered categorical.

meanstructure Logical, should the meanstructure be modeled explicitly?

corinput Logical, is the input a correlation matrix?

fullFIML Logical, should row-wise FIML be used? Not recommended!

Details
The model used in this family is:
\[
\text{var}(y) = \Sigma \\
\mathcal{E}(y) = \mu
\]
in which the covariance matrix can further be modeled in three ways. With type = "chol" as Cholesky decomposition:
\[
\Sigma = LL^T
\]
with type = "prec" as Precision matrix:
\[
\Sigma = K^{-1}
\]
and finally with type = "ggm" as Gaussian graphical model:
\[
\Sigma = \Delta(I - \Omega)^{-1})\Delta.
\]

Value
An object of the class psychonetrics

Author(s)
Sacha Epskamp

References

See Also
lvm, var1, dlvm1

Examples
# Load bfi data from psych package:
library("psychTools")
data(bfi)

# Also load dplyr for the pipe operator:
library("dplyr")

# Let's take the agreeableness items, and gender:
ConsData <- bfi %>%
  select(A1:A5, gender) %>%
  na.omit # Let's remove missingness (otherwise use Estimator = "FIML")

# Define variables:
vars <- names(ConsData)[1:5]

# Saturated estimation:
mod_saturated <- ggm(ConsData, vars = vars)

# Run the model:
mod_saturated <- mod_saturated %>% runmodel

# We can look at the parameters:
mod_saturated %>% parameters

# Labels:
labels <- c(
  "indifferent to the feelings of others",
  "inquire about others' well-being",
  "comfort others",
  "love children",
  "make people feel at ease")

# Plot CIs:
CIplot(mod_saturated, "omega", labels = labels, labelstart = 0.2)

# We can also fit an empty network:
mod0 <- ggm(ConsData, vars = vars, omega = "empty")

# Run the model:
mod0 <- mod0 %>% runmodel

# We can look at the modification indices:
mod0 %>% MIs

# To automatically add along modification indices, we can use stepup:
mod1 <- mod0 %>% stepup

# Let's also prune all non-significant edges to finish:
mod1 <- mod1 %>% prune

# Look at the fit:
mod1 %>% fit

# Compare to original (baseline) model:
compare(baseline = mod0, adjusted = mod1)

# We can also look at the parameters:
mod1 %>% parameters
# Or obtain the network as follows:
getmatrix(mod1, "omega")
Index

* classes
  psychonetrics-class, 58
* datasets
  Jonas, 28
  StarWars, 64

addfit (psychonetrics_update), 60
addMIs (psychonetrics_update), 60
addSEs (psychonetrics_update), 60

bifactor, 5
bootstrap, 6, 16, 34, 51
changedata, 6
cholesky (varcov), 78
CIplot, 7
compare, 9
corr (varcov), 78
cov, 7, 15
covML, 10
covMLtoUB (covML), 10
covUBtoML (covML), 10

diagonalizationMatrix
  (duplicationMatrix), 17
dlvm1, 4, 11, 18, 19, 47, 51, 52, 76, 81
duplicationMatrix, 17

eliminationMatrix (duplicationMatrix), 17
emergencystart, 18
esa, 18
esa_manual (esa), 18

factorscores, 19
fit, 20, 53, 66
fixpar, 21
freepar (fixpar), 21

generate, 22
getmatrix, 22

getVCOV, 24
ggm (varcov), 78
groupequal, 24
groupfree (groupequal), 24
gvar (var1), 73

identify (psychonetrics_update), 60
intersectionmodel (unionmodel), 72
Ising, 25
Jonas, 28

latentgrowth, 29
lnm (lvm), 31
lrnm (lvm), 31
lvm, 4, 5, 30, 31, 47, 76, 81

meta_ggm (meta_varcov), 42
meta_varcov, 42
MIs, 45
ml_gvar (ml_tsdlvm1), 51
ml_lnm (ml_lvm), 47
ml_lrnm (ml_lvm), 47
ml_lvm, 47
ml_rnm (ml_lvm), 47
ml_ts_lvgvar (ml_tsdlvm1), 51
ml_tsdlvm1, 51
ml_var (ml_tsdlvm1), 51
modelsearch, 52

panel_lvgvar (dlvm1), 11
panelgvar (dlvm1), 11
panelvar (dlvm1), 11
parameters, 54
parequal, 55
partialprune, 56
plot.esa (esa), 18
plot.esa_manual (esa), 18
prec (varcov), 78
precision (varcov), 78
print.esa (esa), 18
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>print.esa_manual(esa)</td>
<td>18</td>
</tr>
<tr>
<td>print.psychonetrics_compare(compare)</td>
<td>9</td>
</tr>
<tr>
<td>prune</td>
<td>53, 56, 57, 67</td>
</tr>
<tr>
<td>psychonetrics (psychonetrics-package)</td>
<td>3</td>
</tr>
<tr>
<td>psychonetrics-class</td>
<td>5-7, 16, 21, 25, 30, 35, 45, 51, 53, 55, 57, 58, 60, 61, 63, 64, 66, 71, 73</td>
</tr>
<tr>
<td>psychonetrics-package</td>
<td>3</td>
</tr>
<tr>
<td>psychonetrics_update</td>
<td>60</td>
</tr>
<tr>
<td>resid,psychonetrics-method</td>
<td>58</td>
</tr>
<tr>
<td>residuals,psychonetrics-method</td>
<td>58</td>
</tr>
<tr>
<td>rnm (lvm)</td>
<td>31</td>
</tr>
<tr>
<td>runmodel</td>
<td>53, 57, 61, 66</td>
</tr>
<tr>
<td>setestimator</td>
<td>62</td>
</tr>
<tr>
<td>setoptimizer (setestimator)</td>
<td>62</td>
</tr>
<tr>
<td>setverbose</td>
<td>63</td>
</tr>
<tr>
<td>show,psychonetrics-method</td>
<td>58</td>
</tr>
<tr>
<td>simplestructure</td>
<td>48, 64</td>
</tr>
<tr>
<td>StarWars</td>
<td>64</td>
</tr>
<tr>
<td>stepup</td>
<td>53, 58, 65</td>
</tr>
<tr>
<td>ts_lvgvar(tsdlvml)</td>
<td>67</td>
</tr>
<tr>
<td>tsdlvml</td>
<td>4, 67</td>
</tr>
<tr>
<td>unionmodel</td>
<td>72</td>
</tr>
<tr>
<td>usecpp (setestimator)</td>
<td>62</td>
</tr>
<tr>
<td>var1</td>
<td>4, 73, 81</td>
</tr>
<tr>
<td>varcov</td>
<td>4, 76, 78</td>
</tr>
</tbody>
</table>