Package 'semPower'

August 24, 2020

Type Package

Title Power Analyses for SEM

Version 1.1.0

Author Morten Moshagen

Maintainer Morten Moshagen <morten.moshagen@uni-ulm.de>

Description Provides a-priori, post-hoc, and compromise power-analyses for structural equation models (SEM). Moshagen & Erdfelder (2016) <doi:10.1080/10705511.2014.950896>.

License LGPL

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports stats, grDevices, graphics

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-08-24 09:50:02 UTC

R topics documented:

`checkBounded` *checkBounded*

Description

checks whether x is defined and lies within the specified bound

Usage

```
checkBounded(x, message = NULL, bound = c(0, 1))
```

Arguments

`x`	x
`message`	identifier for x
`bound`	the boundaries, array of size two

checkPositive *checkPositive*

Description

checks whether x is defined and a positive number, stop otherwise

Usage

```
checkPositive(x, message = NULL)
```

Arguments

x	x
message	identifier for x

checkPositiveDefinite *checkPositiveDefinite*

Description

checks whether x is positive definite

Usage

```
checkPositiveDefinite(x, message = NULL)
```

Arguments

x	x
message	identifier for x

getAGFI.F *getAGFI.F*

Description

calculates AGFI from minimum of the ML-fit-function

Usage

```
getAGFI.F(Fmin, df, p)
```

Arguments

`Fmin`	minimum of the ML-fit-function
`df`	model degrees of freedom
`p`	number of observed variables

Value

AGFI

getBetadiff *getBetadiff*

Description

get squared difference between requested and achieved beta on a logscale

Usage

```
getBetadiff(cN, critChi, logBetaTarget, fmin, df, weights = NULL)
```

Arguments

`cN`	current N
`critChi`	critical chi-square associated with chosen alpha error
`logBetaTarget`	log(desired beta)
`fmin`	minimum of the ML fit function
`df`	the model degrees of freedom
`weights`	sample weights for multiple group models

Value

squared difference requested and achieved beta on a log scale

getCFI.Sigma — *getCFI.Sigma*

Description

calculates CFI given model-implied and observed covariance matrix.

Usage

```
getCFI.Sigma(SigmaHat, S)
```

Arguments

`SigmaHat`	model implied covariance matrix
`S`	observed (or population) covariance matrix

Details

cfi= (f_null - f_hyp) / f_null

Value

CFI

getCFI.Sigma.mgroups — *getCFI.Sigma.mgroups*

Description

calculates CFI given model-implied and observed covariance matrix for multiple group models.

Usage

```
getCFI.Sigma.mgroups(SigmaHat, S, N)
```

Arguments

`SigmaHat`	a list of model implied covariance matrix
`S`	a list of observed (or population) covariance matrix
`N`	a list of group weights

Details

cfi= (f_null - f_hyp) / f_null

Value

CFI

getChiSquare.F *getChiSquare.F*

Description

calculates chis-square from the population minimum of the fit-function

Usage

```
getChiSquare.F(Fmin, n, df)
```

Arguments

`Fmin`	population minimum of the fit-function
`n`	number of observations
`df`	model degrees of freedom

Details

chi = (n-1)*F + df = ncp + df

note that F is the population minimum; using F_hat would give chi = (n-1)*F_hat

Value

NCP

getChiSquare.NCP *getChiSquare.NCP*

Description

calculates chi-square from NCP

Usage

```
getChiSquare.NCP(NCP, df)
```

Arguments

`NCP`	non-centrality parameter
`df`	model degrees of freedom

Details

chi = ncp + df

Value

chiSquare

`getErrorDiff` *getErrorDiff*

Description

determine the squared log-difference between alpha and beta error given a certain chi-square value from central chi-square(df) and a non-central chi-square(df, ncp) distribution.

Usage

```
getErrorDiff(critChiSquare, df, ncp, log.abratio)
```

Arguments

`critChiSquare`	evaluated chi-squared value
`df`	the model degrees of freedom
`ncp`	the non-centrality parameter
`log.abratio`	log(alpha/beta)

Value

squared difference between alpha and beta on a log scale

`getF` *getF calculates minimum of the ML-fit-function from known fit indices*

Description

getF calculates minimum of the ML-fit-function from known fit indices

Usage

```
getF(
  effect,
  effect.measure,
  df = NULL,
  p = NULL,
  SigmaHat = NULL,
  Sigma = NULL
)
```

Arguments

effect	magnitude of effect
effect.measure	measure of effect, one of 'fmin','rmsea','agfi','gfi','mc'
df	model degrees of freedom
p	number of observed varaibles
SigmaHat	model implied covariance matrix
Sigma	population covariance matrix

Value

Fmin

---

getF.AGFI *getF.AGFI*

Description

calculates minimum of the ML-fit-function from AGFI

Usage

```
getF.AGFI(AGFI, df, p)
```

Arguments

AGFI	AGFI
df	model degrees of freedom
p	number of observed variables

Details

F_min = rmsea^2 * df

Value

Fmin

getF.GFI *getF.GFI*

Description

calculates minimum of the ML-fit-function from AGFI

Usage

```
getF.GFI(GFI, p)
```

Arguments

GFI	GFI
p	number of observed variables

Value

Fmin

getF.Mc *getF.Mc*

Description

calculates minimum of the ML-fit-function from Mc

Usage

```
getF.Mc(Mc)
```

Arguments

Mc	Mc

Value

Fmin

---

getF.RMSEA *getF.RMSEA*

Description

calculates minimum of the ML-fit-function from RMSEA

Usage

```
getF.RMSEA(RMSEA, df)
```

Arguments

RMSEA	RMSEA
df	model degrees of freedom

Details

F_min = rmsea^2 * df

Value

Fmin

---

getF.Sigma *getF.Sigma*

Description

calculates minimum of the ML-fit-function given model-implied and observed covariance matrix.

Usage

```
getF.Sigma(SigmaHat, S)
```

Arguments

SigmaHat	model implied covariance matrix
S	observed (or population) covariance matrix

Details

F_min = tr(S

Value

Fmin

getFormattedResults *getFormattedResults*

Description

returned dataframe containing formatted results

Usage

```
getFormattedResults(type, result, digits = 6)
```

Arguments

type	type of power analysis
result	result object (list)
digits	number of significant digits

Value

data.frame

getGFI.F *getGFI.F*

Description

calculates GFI from minimum of the ML-fit-function

Usage

```
getGFI.F(Fmin, p)
```

Arguments

Fmin	minimum of the ML-fit-function
p	number of observed variables

Value

GFI

getIndices.F *getIndices.F*

Description

calculates known indices from minimum of the ML-fit-function

Usage

```
getIndices.F(fmin, df, p = NULL, SigmaHat = NULL, Sigma = NULL, N = NULL)
```

Arguments

fmin	minimum of the ML-fit-function
df	model degrees of freedom
p	number of observed variables
SigmaHat	model implied covariance matrix
Sigma	population covariance matrix
N	list of sample weights

Value

list of indices

getMc.F *getMc.F*

Description

calculates Mc from minimum of the ML-fit-function

Usage

```
getMc.F(Fmin)
```

Arguments

Fmin	minimum of the ML-fit-function

Value

Mc

getNCP *getNCP*

Description

calculates non-centrality parameter from the population minimum of the fit-function

Usage

```
getNCP(Fmin, n)
```

Arguments

Fmin	population minimum of the fit-function
n	number of observations

Details

ncp = (n-1) * F

Value

NCP

getRMSEA.F *getRMSEA.F*

Description

calculates RMSEA from minimum of the ML-fit-function

Usage

```
getRMSEA.F(Fmin, df, nGroups = 1)
```

Arguments

Fmin	minimum of the ML-fit-function
df	model degrees of freedom
nGroups	the number of groups

Details

F_min = rmsea^2 * df

Value

RMSEA

getSRMR.Sigma *getSRMR.Sigma*

Description

calculates SRMR given model-implied and observed covariance matrix.

Usage

```
getSRMR.Sigma(SigmaHat, S)
```

Arguments

`SigmaHat`	model implied covariance matrix
`S`	observed (or population) covariance matrix

Value

SRMR

getSRMR.Sigma.mgroups *getSRMR.Sigma.mgroups*

Description

calculates SRMR given model-implied and observed covariance matrix for multiple group models

Usage

```
getSRMR.Sigma.mgroups(SigmaHat, S, N)
```

Arguments

`SigmaHat`	a list of model implied covariance matrices
`S`	a list of observed (or population) covariance matrices
`N`	a list of group weights

Value

SRMR

semPower — *semPower: Power analyses for structural equation models (SEM).*

Description

semPower allows for performing a-priori, post-hoc, and compromise power-analyses for structural equation models (SEM).

Details

- A-priori power analysis `semPower.aPriori` computes the required N, given an effect, alpha, power, and the model df
- Post-hoc power analysis `semPower.postHoc` computes the achieved power, given an effect, alpha, N, and the model df
- Compromise power analysis `semPower.compromise` computes the implied alpha and power, given an effect, the alpha/beta ratio, N, and the model df

In SEM, the discrepancy between H0 and H1 (the magnitude of effect) refers to the difference in fit between two models. If only one model is defined (which is the default), power refers to the global chi-square test. If both models are explicitly defined, power is computed for nested model tests. semPower allows for expressing the magnitude of effect by one of the following measures: F0, RMSEA, Mc, GFI, or AGFI.

Alternatively, the implied effect can also be computed from the discrepancy between the population (or a certain model-implied) covariance matrix defining H0 and the hypothesized (model-implied) covariance matrix from a nested model defining H1. See the examples below how to use this feature in conjunction with lavaan.

Author(s)

Morten Moshagen <morten.moshagen@uni-ulm.de>

semPower.aPriori — *semPower.aPriori*

Description

Determine required sample size given alpha, beta/power, df, and effect

Usage

```
semPower.aPriori(
  effect = NULL,
  effect.measure = NULL,
  alpha,
  beta = NULL,
```

```
  power = NULL,
  N = NULL,
  df,
  p = NULL,
  SigmaHat = NULL,
  Sigma = NULL
)
```

Arguments

effect	effect size specifying the discrepancy between H0 and H1 (a list for multiple group models)
effect.measure	type of effect, one of "F0", "RMSEA", "Mc", "GFI", AGFI"
alpha	alpha error
beta	beta error; set either beta or power
power	power (1-beta); set either beta or power
N	a list of sample weights for multiple group power analyses, e.g. list(1,2) to make the second group twice as large as the first one
df	the model degrees of freedom
p	the number of observed variables, required for effect.measure = "GFI" and "AGFI"
SigmaHat	model implied covariance matrix (a list for multiple group models). Use in conjuntion with Sigma to define effect and effect.measure.
Sigma	population covariance matrix (a list for multiple group models). Use in conjuntion with SigmaHat to define effect and effect.measure.

Value

list

Examples

```
## Not run:
power <- semPower.aPriori(effect = .05, effect.measure = "RMSEA", alpha = .05, beta = .05, df = 200)
power
power <- semPower.aPriori(effect = .15, effect.measure = "F0", alpha = .05, power = .80, df = 100)
power
power <- semPower.aPriori(effect = list(.05, .10), effect.measure = "F0", alpha = .05,
                          power = .80, N = list(1, 1), df = 100)
power
power <- semPower.aPriori(alpha = .01, beta = .05, df = 5,
                        SigmaHat = diag(4), Sigma = cov(matrix(rnorm(4*1000),  ncol=4)))
power

## End(Not run)
```

semPower.compromise *sempower.compromise*

Description

Performs a compromise power analysis, i.e. determines the critical chi-square along with the implied alpha and beta, given a specified alpha/beta ratio, effect, N, and df

Usage

```
semPower.compromise(
  effect = NULL,
  effect.measure = NULL,
  abratio = 1,
  N,
  df,
  p = NULL,
  SigmaHat = NULL,
  Sigma = NULL
)
```

Arguments

`effect`	effect size specifying the discrepancy between H0 and H1 (a list for multiple group models)
`effect.measure`	type of effect, one of "F0","RMSEA", "Mc", "GFI", AGFI"
`abratio`	the ratio of alpha to beta
`N`	the number of observations (a list for multiple group models)
`df`	the model degrees of freedom
`p`	the number of observed variables, required for effect.measure = "GammaHat", "GFI", and "AGFI"
`SigmaHat`	model implied covariance matrix (a list for multiple group models). Use in conjuntion with Sigma to define effect and effect.measure.
`Sigma`	population covariance matrix (a list for multiple group models). Use in conjuntion with SigmaHat to define effect and effect.measure.

Value

list

Examples

```
## Not run:
cp.ph <- semPower.compromise(effect = .08, effect.measure = "RMSEA", abratio = 1, N = 250, df = 200)
summary(cp.ph)

## End(Not run)
```

`semPower.postHoc` *semPower.postHoc*

Description

Determine power (1-beta) given alpha, df, and effect

Usage

```
semPower.postHoc(
  effect = NULL,
  effect.measure = NULL,
  alpha,
  N,
  df,
  p = NULL,
  SigmaHat = NULL,
  Sigma = NULL
)
```

Arguments

`effect`	effect size specifying the discrepancy between H0 and H1 (a list for multiple group models)
`effect.measure`	type of effect, one of "F0", "RMSEA", "Mc", "GFI", AGFI"
`alpha`	alpha error
`N`	the number of observations (a list for multiple group models)
`df`	the model degrees of freedom
`p`	the number of observed variables, required for effect.measure = "GammaHat", "GFI", and "AGFI"
`SigmaHat`	model implied covariance matrix (a list for multiple group models). Use in conjuntion with Sigma to define effect and effect.measure.
`Sigma`	population covariance matrix (a list for multiple group models). Use in conjuntion with SigmaHat to define effect and effect.measure.

Value

list

Examples

```
## Not run:
power <- semPower.postHoc(effect = .05, effect.measure = "RMSEA", alpha = .05, N = 250, df = 200)
power
power <- semPower.postHoc(effect = list(.02, .01), effect.measure = "F0",
                          alpha = .05, N = list(250, 350), df = 200)
```

```
power
power <- semPower.postHoc(N = 1000, df = 5, alpha = .05,
                          SigmaHat = diag(4), Sigma = cov(matrix(rnorm(4*1000),  ncol=4)))
power

## End(Not run)
```

---

semPower.powerPlot.byEffect
sempower.powerPlot.byEffect

Description

show a plot showing power as function of N for a given effect and alpha

Usage

```
semPower.powerPlot.byEffect(
  effect.measure = NULL,
  alpha,
  N,
  df,
  p = NULL,
  effect.min = NULL,
  effect.max = NULL,
  steps = 50,
  linewidth = 1
)
```

Arguments

effect.measure	type of effect, one of "F0", "RMSEA", "Mc", "GFI", AGFI"
alpha	alpha error
N	the number of observations
df	the model degrees of freedom
p	the number of observed variables, required for effect.measure = "GFI" and "AGFI"
effect.min	minimum effect
effect.max	maximum effect
steps	number of steps
linewidth	linewidth

Value

powerplot

Examples

```
## Not run:
semPower.powerPlot.byEffect(effect.measure = "RMSEA", alpha = .05,
                            N = 500, effect.min = .01, effect.max = .15, df = 200)

## End(Not run)
```

---

semPower.powerPlot.byN

sempower.powerPlot.byN

Description

show a plot showing power as function of N for a given effect and alpha

Usage

```
semPower.powerPlot.byN(
  effect = NULL,
  effect.measure = NULL,
  alpha,
  df,
  p = NULL,
  SigmaHat = NULL,
  Sigma = NULL,
  power.min = alpha,
  power.max = 0.999,
  steps = 50,
  linewidth = 1
)
```

Arguments

effect	effect size specifying the discrepancy between H0 and H1
effect.measure	type of effect, one of "F0", "RMSEA", "Mc", "GFI", AGFI"
alpha	alpha error
df	the model degrees of freedom
p	the number of observed variables, required for effect.measure = "GFI" and "AGFI"
SigmaHat	model implied covariance matrix. Use in conjuntion with Sigma to define effect and effect.measure.
Sigma	population covariance matrix. Use in conjuntion with SigmaHat to define effect and effect.measure.
power.min	minimum power, must not be smaller than alpha
power.max	maximum power
steps	number of steps
linewidth	linewidth

Value

powerplot

Examples

```
## Not run:
semPower.powerPlot.byN(effect = .05, effect.measure = "RMSEA",
                       alpha = .05, power.min = .05, power.max = .999, df = 200)

## End(Not run)
```

`semPower.showPlot` *semPower.showPlot*

Description

show a plot showing central and non-central chi-square distribution

Usage

```
semPower.showPlot(chiCrit, ncp, df, linewidth = 1)
```

Arguments

`chiCrit`	critical chi-square, e.g. qchisq(alpha, df, ncp=0, lower.tail = F)
`ncp`	non-centrality parameter under H1
`df`	degrees of freedom
`linewidth`	linewidth

`summary.semPower.aPriori` *summary.semPower.aPriori*

Description

provide summary of a-priori power analyses

Usage

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

Arguments

`object`	result object from semPower.aPriori
`...`	other

summary.semPower.compromise *summary.sempower.compromise*

Description

provide summary of compromise post-hoc power analyses

Usage

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

Arguments

`object`	result object from semPower.compromise
`...`	other

summary.semPower.postHoc *semPower.postHoc.summary*

Description

provide summary of post-hoc power analyses

Usage

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

Arguments

`object`	result object from semPower.posthoc
`...`	other

validateInput *validateInput*

Description

Validates input for power calcuation function

Usage

```
validateInput(
  power.type = NULL,
  effect = NULL,
  effect.measure = NULL,
  alpha = NULL,
  beta = NULL,
  power = NULL,
  abratio = NULL,
  N = NULL,
  df = NULL,
  p = NULL,
  SigmaHat = NULL,
  Sigma = NULL,
  power.min = alpha,
  power.max = 0.999,
  effect.min = NULL,
  effect.max = NULL,
  steps = 50,
  linewidth = 1
)
```

Arguments

`power.type`	type of power analyses, one of "a-priori", post-hoc", "compromise", "powerplot.byN", "powerplot.byEffect"
`effect`	effect size specifying the discrepancy between H0 and H1
`effect.measure`	type of effect, one of "F0, "RMSEA", "Mc", "GFI", AGFI"
`alpha`	alpha error
`beta`	beta error
`power`	power (1-beta)
`abratio`	ratio alpha/beta
`N`	the number of observations
`df`	the model degrees of freedom
`p`	the number of observed variables, required for effect.measure = "GFI" and "AGFI"
`SigmaHat`	model implied covariance matrix

`Sigma`	population covariance matrix
`power.min`	for plotting: minimum power
`power.max`	for plotting: maximum power
`effect.min`	for plotting: minimum effect
`effect.max`	for plotting: maximum effect
`steps`	for plotting: number of sampled points
`linewidth`	for plotting: linewidth

Index