Package ‘semPower’

May 17, 2020

Type Package
Title Power Analyses for SEM
Version 1.0.1
Author Morten Moshagen
Maintainer Morten Moshagen <morten.moshagen@uni-ulm.de>
License LGPL
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0
Imports stats, grDevices, graphics
Suggests knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2020-05-17 07:10:02 UTC

R topics documented:

  checkBounded .................................................. 2
  checkPositive .................................................. 3
  checkPositiveDefinite ......................................... 3
  getAGFLF ....................................................... 4
  getBetadiff ..................................................... 4
  getCFL.Sigma ................................................... 5
  getChiSquare.F ................................................ 5
  getChiSquare.NCP ............................................. 6
  getErrorDiff .................................................. 6
  getF ............................................................ 7
  getF.AGFI ....................................................... 8
checkBounded

Description

checks whether x is defined and lies within the specified bound

Usage

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

Arguments

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

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

Usage
checkPositive(x, message)

Arguments
x x
message identifier for x

checkPositiveDefinite

Description
checks whether x is positive definite

Usage
checkPositiveDefinite(x, message)

Arguments
x x
message identifier for x
### getAGFI.F

**Description**

calculates AGFI from minimum of the ML-fit-function

**Usage**

```r
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

**Description**

get squared difference between requested and achieved beta on a logscale

**Usage**

```r
getBetadiff(cN, critChi, logBetaTarget, fmin, df)
```

**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

**Value**

squared difference requested and achieved beta on a log scale
### Description

Calculates CFI given model-implied and observed covariance matrix.

### Usage

```r
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 \( F_{\text{hat}} \) would give \( \chi = (n-1)*F_{\text{hat}} \).
**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**

**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**

ggetErrorDiff(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)


\textit{getF} calculates minimum of the ML-fit-function from known fit indices

\textbf{Description}

\textit{getF} calculates minimum of the ML-fit-function from known fit indices

\textbf{Usage}

\begin{verbatim}
getF(
effect, 
effect.measure,  
\text{df} = \text{NULL},  
\text{p} = \text{NULL},  
\text{SigmaHat} = \text{NULL},  
\text{Sigma} = \text{NULL}
)
\end{verbatim}

\textbf{Arguments}

- \textit{effect} : magnitude of effect
- \textit{effect.measure} : measure of effect, one of 'fmin','rmsea','agfi','gfi','mc'
- \textit{df} : model degrees of freedom
- \textit{p} : number of observed variables
- \textit{SigmaHat} : model implied covariance matrix
- \textit{Sigma} : population covariance matrix

\textbf{Value}

- Fmin
**getF.AGFI**

**Description**

calculates minimum of the ML-fit-function from AGFI

**Usage**

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

**Arguments**

- `AGFI`: AGFI
- `df`: model degrees of freedom
- `p`: number of observed variables

**Details**

\[ F_{\text{min}} = \text{rmsea}^2 \times df \]

**Value**

- `Fmin`

---

**getF.GFI**

**Description**

calculates minimum of the ML-fit-function from AGFI

**Usage**

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

**Arguments**

- `GFI`: GFI
- `p`: number of observed variables

**Value**

- `Fmin`
**getF.Mc**

**Description**
calculates minimum of the ML-fit-function from Mc

**Usage**
getF.Mc(Mc)

**Arguments**
- **Mc**

**Value**
Fmin

---

**getF.RMSEA**

**Description**
calculates minimum of the ML-fit-function from RMSEA

**Usage**
getF.RMSEA(RMSEA, df)

**Arguments**
- **RMSEA**
- **df**
  - model degrees of freedom

**Details**
F_min = rmsea^2 * df

**Value**
Fmin
**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**

\[ F_{\text{min}} = \text{tr}(S) \]

**Value**

`Fmin`

---

**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

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

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

Usage
getIndices.F(fmin, df, p = NULL, SigmaHat = NULL, Sigma = 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

Value
list of indices
getMc.\(F\)

**Description**

calculates \(M_c\) from minimum of the ML-fit-function

**Usage**

\[
\text{getMc.}\(F\)(\(F_{\text{min}}\))
\]

**Arguments**

\(F_{\text{min}}\) minimum of the ML-fit-function

**Value**

\(M_c\)

---

getNCP

**Description**

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

**Usage**

\[
\text{getNCP}(F_{\text{min}}, n)
\]

**Arguments**

\(F_{\text{min}}\) population minimum of the fit-function
\(n\) number of observations

**Details**

\[
\text{ncp} = (n-1) \times F
\]

**Value**

\(NCP\)
getRMSEA.F

Description

Calculates RMSEA from minimum of the ML-fit-function

Usage

getRMSEA.F(Fmin, df)

Arguments

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

Details

\[ F_\text{min} = \text{rmsea}^2 \times df \]

Value

RMSEA

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
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

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,
df,
p = NULL,
SigmaHat = NULL,
Sigma = NULL)

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
beta beta error; set either beta or power
power power (1-beta); set either beta or power
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 conjunction with Sigma to define effect and effect.measure.
Sigma population covariance matrix. Use in conjunction 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(alpha = .01, beta = .05, df = 5,
                           SigmaHat = diag(4), Sigma = cov(matrix(rnorm(4*1000), ncol=4)))
power

## End(Not run)
semPower.postHoc

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
effect.measure  type of effect, one of "F0", "RMSEA", "Mc", "GFI", AGFI"
abratio         the ratio of alpha to beta
N                the number of observations
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. Use in conjunction with Sigma to define effect
                 and effect.measure.
Sigma            population covariance matrix. Use in conjunction 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

Description

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

```r
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

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 = "GammaHat", "GFI", and "AGFI"

SigmaHat: model implied covariance matrix. Use in conjunction with Sigma to define effect and effect.measure.

Sigma: population covariance matrix. Use in conjunction with SigmaHat to define effect and effect.measure.

Value

list

Examples

```r
## Not run:
power <- semPower.postHoc(effect = .05, effect.measure = "RMSEA", alpha = .05, N = 250, 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)
```
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

typeofpowerplot

Examples

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

## End(Not run)
Description

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

Usage

```r
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 conjunction with Sigma to define effect and effect.measure.
- `Sigma`: population covariance matrix. Use in conjunction 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

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

Description

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

Usage

```r
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

Description

provide summary of a-priori power analyses

Usage

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

Arguments

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

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

Description
Validates input for power calculation 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
validateInput

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigma</td>
<td>Population covariance matrix</td>
</tr>
<tr>
<td>power.min</td>
<td>For plotting: minimum power</td>
</tr>
<tr>
<td>power.max</td>
<td>For plotting: maximum power</td>
</tr>
<tr>
<td>effect.min</td>
<td>For plotting: minimum effect</td>
</tr>
<tr>
<td>effect.max</td>
<td>For plotting: maximum effect</td>
</tr>
<tr>
<td>steps</td>
<td>For plotting: number of sampled points</td>
</tr>
<tr>
<td>linewidth</td>
<td>For plotting: linewidth</td>
</tr>
</tbody>
</table>
### Index

- checkBounded, 2
- checkPositive, 3
- checkPositiveDefinite, 3
- getAGFI.F, 4
- getBetadiff, 4
- getCFI.Sigma, 5
- getChiSquare.F, 5
- getChiSquare.NCP, 6
- getErrorDiff, 6
- getF, 7
- getF.AGFI, 8
- getF.GFI, 8
- getF.Mc, 9
- getF.RMSEA, 9
- getF.Sigma, 10
- getFormattedResults, 10
- getGFI.F, 11
- getIndices.F, 11
- getMc.F, 12
- getNCP, 12
- getRMSEA.F, 13
- getSRMR.Sigma, 13
- semPower, 14
- semPower-package (semPower), 14
- semPower.aPriori, 14
- semPower.compromise, 14
- semPower.postHoc, 14
- semPower.powerPlot.byEffect, 18
- semPower.powerPlot.byN, 19
- semPower.showPlot, 20
- summary.semPower.aPriori, 20
- summary.semPower.compromise, 21
- summary.semPower.postHoc, 21
- validateInput, 22