Package ‘semPower’

March 27, 2018

Type Package
Title Power Analyses for SEM
Version 1.0.0
Author Morten Moshagen
Maintainer Morten Moshagen <morten.moshagen@uni-ulm.de>
License LGPL
Encoding UTF-8
LazyData true
RoxygenNote 6.0.1
Imports stats, grDevices, graphics
Suggests knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2018-03-27 18:08:46 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 ........................................................ 7
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
- message: identifier for x

---

**checkPositiveDefinite**

**Description**

checks whether x is positive definite

**Usage**

checkPositiveDefinite(x, message)

**Arguments**

- x
- message: identifier for x
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

Description
get squared difference between requested and achieved beta on a logscale

Usage
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
**getCFI.Sigma**

**Description**

Calculates CFI given model-implied and observed covariance matrix.

**Usage**

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

**Arguments**

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

**Details**

\[ \text{cfi} = \frac{(f_{\text{null}} - f_{\text{hyp}})}{f_{\text{null}}} \]

**Value**

CFI

**getChiSquare.F**

**Description**

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

**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**

\[ \text{chi} = (n-1)*F + df = ncp + df \]

Note that F is the population minimum; using F_hat would give \( \text{chi} = (n-1)*F_{\text{hat}} \)
**getErrorDiff**

**Value**

NCP

---

**getDescription**

`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

---

**getDescription**

`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)
getF

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 variables
SigmaHat model implied covariance matrix
Sigma population covariance matrix

Value

Fmin

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

generated image
### Details

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

### Value

- **getFMc**

### Description

- **getF.GFI**

  calculates minimum of the ML-fit-function from AGFI

### Usage

- `getF.GFI(GFI, p)`

### Arguments

- **GFI**
  - GFI
- **p**
  - number of observed variables

### Value

- **Fmin**

### Description

- **getF.Mc**

  calculates minimum of the ML-fit-function from Mc

### Usage

- `getF.Mc(Mc)`

### Arguments

- **Mc**
  - Mc

### Value

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

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

Usage

getMc.F(Fmin)

Arguments

- `Fmin`: minimum of the ML-fit-function

Value

Mc
getNCPrmseaR

**getNCPrmseaR**

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

**Usage**
getNCPrmseaR(Fmin, n)

**Arguments**
- `Fmin`: population minimum of the fit-function
- `n`: number of observations

**Details**
ncp = (n-1) * F

**Value**
NCP

getRMSEAr

**getRMSEAr**

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

**Usage**
getRMSEAr(Fmin, df)

**Arguments**
- `Fmin`: minimum of the ML-fit-function
- `df`: model degrees of freedom

**Details**
F_min = rmsea^2 * 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

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.
Examples

# a-priori power analyses using rmsea = .05 a target power (1-beta) of .80
ap1 <- semPower.aPriori(0.05, 'RMSEA', alpha = .05, beta = .20, df = 200)
summary(ap1)

# a-priori power analyses using f0 = .75 and a target power of .95
ap2 <- semPower.aPriori(0.75, 'F0', alpha = .05, power = .95, df = 200)
summary(ap2)

# create a plot showing how power varies by N (given a certain effect)
semPower.powerPlot.byN(.05, 'RMSEA', alpha=.05, df=200, power.min=.05, power.max=.99)

# post-hoc power analyses using rmsea = .08
ph <- semPower.postHoc(.08, 'RMSEA', alpha = .05, N = 250, df = 50)
summary(ph)

# create a plot showing how power varies by the magnitude of effect (given a certain N)
semPower.powerPlot.byEffect('RMSEA', alpha=.05, N = 100, df=200, effect.min=.001, effect.max=.10)

# compromise power analyses using rmsea = .08 and an abratio of 2
cp <- semPower.compromise(.08, 'RMSEA', abratio = 2, N = 1000, df = 200)
summary(cp)

# use lavaan to define effect through covariance matrices:
## Not run:
library(lavaan)

# define population model (= H0)
model.pop <- 
  f1 =~ .8*x1 + .7*x2 + .6*x3
  f2 =~ .7*x4 + .6*x5 + .5*x6
  f1 =~ 1*f1
  f2 =~ 1*f2
  f1 =~ 0.5*f2

# define (wrong) H1 model
model.h1 <- 
  f1 =~ x1 + x2 + x3
  f2 =~ x4 + x5 + x6
  f1 =~ 0*f2

# get population covariance matrix; equivalent to a perfectly fitting H0 model
cov.h0 <- fitted(sem(model.pop))$cov

# get covariance matrix as implied by H1 model
res.h1 <- sem(model.h1, sample.cov = cov.h0, sample.nobs = 1000, likelihood='wishart')
df <- res.h1@test[11]$df
cov.h1 <- fitted(res.h1)$cov
## do power analyses

# post-hoc
ph <- semPower.postHoc(SigmaHat = cov.h1, Sigma = cov.h0, alpha = .05, N = 1000, df = df)
summary(ph)
# => Power to reject the H1 model is > .9999 (1-beta = 1-1.347826e-08) with N = 1000 at alpha=.05

# compare:
ph$fmin * (ph$N-1)
fitmeasures(res.h1, 'chisq')
# => expected chi-square matches empirical chi-square

# a-priori
ap <- semPower.aPriori(SigmaHat = cov.h1, Sigma = cov.h0, alpha = .05, power = .80, df = df)
summary(ap)
# => N = 194 gives a power of ~80% to reject the H1 model at alpha = .05

# compromise
cp <- semPower.compromise(SigmaHat = cov.h1, Sigma = cov.h0, abratio = 1, N = 1000, df = df)
summary(cp)
# => A critical Chi-Squared of 33.999 gives balanced alpha-beta
#    error probabilities of alpha=beta=0.000089 with N = 1000.

## End(Not run)

---

### 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, 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

---
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**

```r
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"

**Value**

list

**Examples**

```r
## 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)
```
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:
cp.ph <- semPower.compromise(effect = .08, effect.measure = "RMSEA", abratio = 1, N = 250, df = 200)
summary(cp.ph)

## End(Not run)
```
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

```r
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

```r
## 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

`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 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

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

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

Arguments

object result object from semPower.aPriori
... other
**summary.semPower.compromise**

**Description**

provide summary of compromise post-hoc power analyses

**Usage**

```r
## 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**

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

**Arguments**

- `object`: result object from semPower.posthoc
- `...`: other
validateInput

validateInput

---

**Description**

Validates input for power calculation function

**Usage**

```r
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", "power-plot.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

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, 7
getF.GFI, 8
getF.Mc, 8
getF.RMSEA, 9
getF.Sigma, 9
getFormattedResults, 10
getGFI.F, 10
getIndices.F, 11
getMc.F, 11
getNCP, 12
getRMSEA.F, 12
getSRMR.Sigma, 13

semPower, 13
semPower-package (semPower), 13
semPower.aPriori, 13, 15
semPower.compromise, 13, 16
semPower.postHoc, 13, 17
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