

# Package ‘semPower’

October 14, 2022

**Type** Package

**Title** Power Analyses for SEM

**Version** 1.2.0

**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

**URL** <https://github.com/moshagen/semPower>

**BugReports** <https://github.com/moshagen/semPower/issues>

**Imports** graphics, grDevices, stats, utils

**Suggests** knitr, rmarkdown, lavaan

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2021-11-01 23:10:02 UTC

## R topics documented:

checkBounded . . . . .	2
checkPositive . . . . .	3
checkPositiveDefinite . . . . .	3
checkPowerTypes . . . . .	4
getAGFI.F . . . . .	4
getBetadiff . . . . .	5
getCFI.Sigma . . . . .	5
getCFI.Sigma.mgroups . . . . .	6

getChiSquare.F . . . . .	6
getChiSquare.NCP . . . . .	7
getErrorDiff . . . . .	7
getF . . . . .	8
getF.AGFI . . . . .	9
getF.GFI . . . . .	9
getF.Mc . . . . .	10
getF.RMSEA . . . . .	10
getF.Sigma . . . . .	11
getFormattedResults . . . . .	11
getGFI.F . . . . .	12
getIndices.F . . . . .	12
getMc.F . . . . .	13
getNCP . . . . .	13
getRMSEA.F . . . . .	14
getSRMR.Sigma . . . . .	14
getSRMR.Sigma.mgroups . . . . .	15
semPower . . . . .	15
semPower.aPriori . . . . .	18
semPower.compromise . . . . .	20
semPower.getDf . . . . .	21
semPower.postHoc . . . . .	21
semPower.powerCFA . . . . .	23
semPower.powerPlot.byEffect . . . . .	26
semPower.powerPlot.byN . . . . .	27
semPower.showPlot . . . . .	28
summary.semPower.aPriori . . . . .	28
summary.semPower.compromise . . . . .	29
summary.semPower.postHoc . . . . .	29
validateInput . . . . .	30

<b>Index</b>	<b>32</b>
--------------	-----------

---

checkBounded	<i>checkBounded</i>
--------------	---------------------

---

## Description

checks whether x is defined and lies within the specified bound

## Usage

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

**Arguments**

x	x
message	identifier for x
bound	the boundaries, array of size two
inclusive	whether x might lie on boundary

---

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

checkPowerTypes      *checkPowerTypes*

---

**Description**

checks whether type is one of 'a-priori', 'post-hoc', or 'compromise'.

**Usage**

checkPowerTypes(type)

**Arguments**

type                  type

**Value**

valid type

---

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	<i>getBetadiff</i>
-------------	--------------------

---

**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	<i>getCFI.Sigma</i>
--------------	---------------------

---

**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 chi-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**

$\text{chi} = \text{ncp} + \text{df}$

**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	<i>getF calculates minimum of the ML-fit-function from known fit indices</i>
------	--

---

**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`*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**
$$F_{\min} = \text{rmsea}^2 * \text{df}$$
**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**

$$F_{\min} = \text{rmsea}^2 * \text{df}$$
**Value**

Fmin

---

getF.Sigma	<i>getF.Sigma</i>
------------	-------------------

---

**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**

$$F_{\min} = \text{tr}(S)$$
**Value**

Fmin

---

getFormattedResults	<i>getFormattedResults</i>
---------------------	----------------------------

---

**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).

Perform a power analysis. This is a wrapper function for a-priori, post-hoc, and compromise power analyses.

### Usage

```
semPower(type, ...)
```

### Arguments

type	type of power analysis, one of 'a-priori', 'post-hoc', 'compromise'
...	other parameters related to the specific type of power analysis requested

## 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`.

## Value

list

## Author(s)

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

## See Also

Useful links:

- <https://github.com/moshagen/semPower>
- Report bugs at <https://github.com/moshagen/semPower/issues>

[`semPower.aPriori()`], [`semPower.postHoc()`], [`semPower.compromise()`]

## 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)
# generic version
gap1 <- semPower(type = 'a-priori', 0.05, 'RMSEA', alpha = .05, beta = .20, df = 200)
summary(gap1)
# 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)
# generic version
gph1 <- semPower(type = 'post-hoc', .08, 'RMSEA', alpha = .05, N = 250, df = 50)
summary(gph1)
# 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)
# generic version
gcp <- semPower(type = 'compromise', .08, 'RMSEA', abratio = 2, N = 1000, df = 200)
summary(gcp)

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

# define population model (= H1)
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) H0 model
model.h0 <- '
f1 =~ x1 + x2 + x3
f2 =~ x4 + x5 + x6
f1 ~~ 0*f2
'

# get population covariance matrix; equivalent to a perfectly fitting H1 model
cov.h1 <- fitted(sem(model.pop))$cov
# get covariance matrix as implied by H0 model
res.h0 <- sem(model.h0, sample.cov = cov.h1, sample.nobs = 1000,
              likelihood='wishart', sample.cov.rescale = F)
df <- res.h0@test[[1]]$df
cov.h0 <- fitted(res.h0)$cov

## do power analyses

# post-hoc
ph <- semPower.postHoc(SigmaHat = cov.h0, Sigma = cov.h1, 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.h0, Sigma = cov.h1, 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.h0, Sigma = cov.h1, 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)

## Not run:

ap <- semPower(type = 'a-priori',
               effect = .08, effect.measure = "RMSEA",
               alpha = .05, beta = .05, df = 200)
summary(ph)

ph <- semPower(type = 'post-hoc',
               effect = .08, effect.measure = "RMSEA",
               alpha = .05, N = 250, df = 200)
summary(ph)

cp <- semPower(type = 'compromise',
               effect = .08, effect.measure = "RMSEA",
               abratio = 1, N = 250, df = 200)
summary(ph)

## 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,
  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 conjunction with Sigma to define effect and effect.measure.
Sigma	population covariance matrix (a list for multiple group models). 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)
summary(power)
power <- semPower.aPriori(effect = .15, effect.measure = "F0", alpha = .05, power = .80, df = 100)
summary(power)
power <- semPower.aPriori(effect = list(.05, .10), effect.measure = "F0", alpha = .05,
                          power = .80, N = list(1, 1), df = 100)
summary(power)
power <- semPower.aPriori(alpha = .01, beta = .05, df = 5,
                          SigmaHat = diag(4), Sigma = cov(matrix(rnorm(4*1000), ncol=4)))
summary(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 = "GFI" and "AGFI"
SigmaHat	model implied covariance matrix (a list for multiple group models). Use in conjunction with Sigma to define effect and effect.measure.
Sigma	population covariance matrix (a list for multiple group models). 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.getDf	<i>semPower.getDf</i>
----------------	-----------------------

---

### Description

Convenience function to determine the degrees of freedom of a given model provided as lavaan model string. This requires the lavaan package.

### Usage

```
semPower.getDf(lavModel)
```

### Arguments

lavModel      the lavaan model string

### Value

df

### Examples

```
## Not run:  
lavModel <- '  
f1 =~ x1 + x2 + x3 + x4  
f2 =~ x5 + x6 + x7 + x8  
f3 =~ y1 + y2 + y3  
f3 ~ f1 + f2  
'  
semPower.getDf(lavModel)  
  
## End(Not run)
```

---

semPower.postHoc	<i>semPower.postHoc</i>
------------------	-------------------------

---

### 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 = "GFI" and "AGFI"
SigmaHat	model implied covariance matrix (a list for multiple group models). Use in conjunction with Sigma to define effect and effect.measure.
Sigma	population covariance matrix (a list for multiple group models). Use in conjunction 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)
summary(power)
power <- semPower.postHoc(effect = list(.02, .01), effect.measure = "F0",
  alpha = .05, N = list(250, 350), df = 200)
summary(power)
power <- semPower.postHoc(N = 1000, df = 5, alpha = .05,
  SigmaHat = diag(4), Sigma = cov(matrix(rnorm(4*1000), ncol=4)))
summary(power)

## End(Not run)
```

---

 semPower.powerCFA      *semPower.powerCFA*


---

### Description

Convenience function for performing power analysis for simple CFA models involving one hypothesized zero correlation between factors. This requires the lavaan package.

### Usage

```
semPower.powerCFA(
  type,
  comparison = "restricted",
  phi,
  nullCor = NULL,
  lambda = NULL,
  nIndicator = NULL,
  loadM = NULL,
  loadSD = NULL,
  loadMinMax = NULL,
  ...
)
```

### Arguments

type	type of power analysis, one of 'a-priori', 'post-hoc', 'compromise'
comparison	comparison model, one of 'saturated' or 'restricted'. This determines the df for power analyses. 'Saturated' provides power to reject the model when compared to the saturated model, so the df equal the one of the hypothesized model. 'Restricted' provides power to reject the model when compared to a model that just restricts the parameter defined by nullCor to zero, so the df are always 1.
phi	factor correlation matrix or single number giving correlation between all factors
nullCor	vector of size 2 indicating which factor correlation in phi is hypothesized to equal zero, e.g. c(1, 2) to refer to the correlation between first and second factor
lambda	a list providing factor loadings by factor. Must not contain secondary loadings.
nIndicator	vector indicating the number of indicators for each factor, e.g. c(4, 6) to define two factors with 4 and 6 indicators, respectively
loadM	vector giving mean loadings for each factor or single number to use for every loading
loadSD	vector giving the standard deviation of loadings for each factor for use in conjunction with loadM. When NULL, SDs are set to zero.
loadMinMax	matrix giving the minimum and maximum loading for each factor or vector to apply to all factors
...	other parameters related to the specific type of power analysis requested

**Value**

a list containing the results of the power analysis, Sigma and SigmaHat as well as several lavaan model strings (modelPop, modelTrue, and modelAna)

**Examples**

```
## Not run:
# a priori power analysis only providing the number of indicators to define
# two factors with correlation of phi and same loading for all indicators
cfapower.ap <- semPower.powerCFA(type = 'a-priori',
                                phi = .2, nIndicator = c(5, 6), loadM = .5,
                                alpha = .05, beta = .05)
                                summary(cfapower.ap$power)

# sanity check: fit true model to population Sigma to evaluate everything was set up as intended
summary(lavaan::sem(cfapower.ap$modelTrue, sample.cov = cfapower.ap$Sigma,
                    sample.nobs = 1000, likelihood = 'wishart', sample.cov.rescale = FALSE),
        stand = TRUE)

# peek into lavaan model strings:
# population model
cfapower.ap$modelPop
# (incorrect) analysis model
cfapower.ap$modelAna

# or plug the population Sigma and model-implied SigmaHat
# into a regular power analysis command
ph <- semPower.aPriori(SigmaHat = cfapower.ap$SigmaHat, Sigma = cfapower.ap$Sigma,
                      df = 1, alpha = .05, beta = .05)
                      summary(ph)

# same as above, but compare to the saturated model
# (rather than to the less restricted model)
#' cfapower.ap <- semPower.powerCFA(type = 'a-priori', comparison = 'saturated',
                                   phi = .2, nIndicator = c(5, 6), loadM = .5,
                                   alpha = .05, beta = .05)

# same as above, but request a compromise power analysis
cfapower.cp <- semPower.powerCFA(type = 'compromise',
                                 phi = .2, nIndicator = c(5, 6), loadM = .5,
                                 abratio = 1, N = 200)

# same as above, but request a post-hoc power analysis
cfapower.ph <- semPower.powerCFA(type = 'post-hoc',
                                 phi = .2, nIndicator = c(5, 6), loadM = .5,
                                 alpha = .05, N = 200)

# post-hoc power analysis providing factor correlation matrix
# and reduced loading matrix
phi <- matrix(c(
  c(1.0, 0.1),
  c(0.1, 1.0)
```



```

    ), byrow = T, ncol = 2)

# lambda: only define primary loadings
# must not contain secondary loadings
lambda <- list(
  c(0.4, 0.5, 0.8),
  c(0.7, 0.6, 0.5, 0.4)
)

cfapower <- semPower.powerCFA(type = 'post-hoc',
  phi = phi, nullCor = c(1, 2), lambda = lambda,
  alpha = .05, N = 250)

# post-hoc power analysis providing factor correlation matrix,
# number of indicators by factor, and min-max loading for all factors
phi <- matrix(c(
  c(1.0, 0.2, 0.5),
  c(0.2, 1.0, 0.3),
  c(0.5, 0.3, 1.0)
), byrow = TRUE, ncol = 3)

cfapower <- semPower.powerCFA(type = 'post-hoc',
  phi = phi, nullCor = c(1, 2),
  nIndicator = c(6, 5, 4), loadMinMax = c(.3, .8),
  alpha = .05, N = 250)

# same as above, but providing mean and sd loading for all factors
cfapower <- semPower.powerCFA(type = 'post-hoc',
  phi = phi, nullCor = c(1, 2),
  nIndicator = c(6, 5, 4), loadM = .5, loadSD = .1,
  alpha = .05, N = 250)

# same as above, but hypothesizing zero correlation between factors 2 and 3
cfapower <- semPower.powerCFA(type = 'post-hoc',
  phi = phi, nullCor = c(2, 3),
  nIndicator = c(6, 5, 4), loadM = .5, loadSD = .1,
  alpha = .05, N = 250)

# same as above, but providing mean and sd of loadings for each factor
cfapower <- semPower.powerCFA(type = 'post-hoc',
  phi = phi, nullCor = c(1, 2),
  nIndicator = c(3, 6, 5),
  loadM = c(.5, .6, .7), loadSD = c(.1, .05, 0),
  alpha = .05, N = 250)

# same as above, but using min-max loadings for each factor
loadMinMax <- matrix(c(
  c(.4, .6),
  c(.5, .8),
  c(.3, .7)
), byrow = TRUE, nrow = 3)

cfapower <- semPower.powerCFA(type = 'post-hoc',

```

```

phi = phi, nullCor = c(1, 2), nIndicator = c(3, 6, 5),
loadMinMax = loadMinMax,
alpha = .05, N = 250)

```

```
## 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

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

```
## 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	<i>validateInput</i>
---------------	----------------------

---

### 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", "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  
checkPowerTypes, 4

getAGFI.F, 4  
getBetadiff, 5  
getCFI.Sigma, 5  
getCFI.Sigma.mgroups, 6  
getChiSquare.F, 6  
getChiSquare.NCP, 7  
getErrorDiff, 7  
getF, 8  
getF.AGFI, 9  
getF.GFI, 9  
getF.Mc, 10  
getF.RMSEA, 10  
getF.Sigma, 11  
getFormattedResults, 11  
getGFI.F, 12  
getIndices.F, 12  
getMc.F, 13  
getNCP, 13  
getRMSEA.F, 14  
getSRMR.Sigma, 14  
getSRMR.Sigma.mgroups, 15

semPower, 15  
semPower-package (semPower), 15  
semPower.aPriori, 16, 18  
semPower.compromise, 16, 20  
semPower.getDf, 21  
semPower.postHoc, 16, 21  
semPower.powerCFA, 23  
semPower.powerPlot.byEffect, 26  
semPower.powerPlot.byN, 27  
semPower.showPlot, 28  
summary.semPower.aPriori, 28  
summary.semPower.compromise, 29  
summary.semPower.postHoc, 29

validateInput, 30