Package: sse (via r-universe)

November 24, 2024

Type Package

Title Sample Size Estimation

Version 0.7-15

Author Thomas Fabbro [aut, cre]

Maintainer Thomas Fabbro <thomas.fabbro@unibas.ch>

URL http://r-forge.r-project.org/projects/power/

BugReports http://r-forge.r-project.org/projects/power/

Description Provides functions to evaluate user-defined power functions for a parameter range, and draws a sensitivity plot. It also provides a resampling procedure for semi-parametric sample size estimation and methods for adding information to a Sweave report.

License GPL-3

LazyLoad yes

Imports methods, grid, lattice, graphics, stats, parallel

Suggests testthat

NeedsCompilation no

Repository https://ctu-bern.r-universe.dev

RemoteUrl https://github.com/thofab/sse

RemoteRef HEAD

RemoteSha 458f5cf27567f622fc577094a57305a5f76553bb

Contents

`Extracting actual elements from objects of class powPar`

Extracting an actual n, theta, and xi

Description

Extracting the actual `n`, `theta`, or `xi` from an object of class powPar. This functions are needed within the 'power-function' for extracting always the actual element during evaluation.

Usage

```
n(x)
theta(x)
xi(x)
```

Arguments

`x`	An object of class powPar.

Details

During the evaluation process with `powCalc` every combination of `n`, `theta`, and `xi` is evaluated. The described functions extract the actual `n`, `theta`, or `xi` during the evaluation process. The evaluation process with `powCalc` changes the actual element to ensure that all combinations are evaluated.

When a objcect of class powPar is created, the first element of `n`, `theta`, or `xi` is also set to be the actual element. This allows to use this method also outside the evaluation with `powCalc` for testing the 'power function'.

Value

An integer value for `n`. A numeric value for `theta` and `xi`.

Note

Do not use the method `pp` inside the power-function e.g. like `pp(x, "n")`, because this would extract the whole vector of `n` and not just the actual element.

See Also

`pp`, for extracting all other elements provided by the user (exept `n`, `theta`, and `xi`.

Examples

```
## defining the range of n and theta to be evaluated
psi <- powPar(n = seq(from = 20, to = 60, by = 2),
              theta = seq(from = 0.5, to = 1.5, by = 0.1),
              muA = 0,
              muB = 1)

## extracting all elements of psi individually, starting with the first
n(psi)
theta(psi)
xi(psi)

## extracting all elements, not just the actual:
pp(psi, name = "n")
pp(psi, name = "theta")
pp(psi, name = "xi")

## an example of usage
powFun <- function(psi){
  power.t.test(n = n(psi),
               delta = pp(psi, "muA") - pp(psi, "muB"),
               sd = theta(psi)
               )$power
}

## testing the power-function
powFun(psi)
```

`Extracting from objects of class powPar`

Extracting from an object of class powPar

Description

All information needed for the 'power-function' should be provided by an object of class `powPar`. To extract this information the function `pp` should be used.

Usage

```
pp(x, name)
```

Arguments

`x`	An object of class powPar.
`name`	A character indicating the name of the object to be extracted.

Value

Everything that can be stored within a list is possible.

Note

The name pp is an abbreviation for power parameter.

See Also

For extracting individual elements of n, theta and xi the functions n, theta, or xi should be used.

Examples

```
psi <- powPar(theta = seq(from = 0.5, to = 1.5, by = 0.5),
              n = seq(from = 10, to = 30, by = 10),
              muA = 0,
              muB = 1)


pp(psi, name = "muA")

## an example of usage
powFun <- function(psi){
  power.t.test(n = n(psi),
               delta = pp(psi, "muA") - pp(psi, "muB"),
               sd = theta(psi)
               )$power
}

## testing the power-function
powFun(psi)
```

---

inspect *Inspection Plot*

Description

Creating a plot that allows to inspect the sample size calculation.

Usage

```
inspect(object)
```

Arguments

object	An object of class power.

Details

The plot shows for every evaluated theta the sample size and the power on a transformed scale. The method used for sample size estimation 'step' or 'lm' is indicated. If the method 'lm' is used a red regression line is shown for the range that was used for estimation.

Value

A plot is generated but nothing is returned.

Examples

```
## defining the range of n and theta to be evaluated
psi <- powPar(theta = seq(from = 0.5, to = 1.5, by = 0.1),
              n = seq(from = 20, to = 60, by = 2),
              muA = 0,
              muB = 1)

## defining a power-function
powFun <- function(psi){
  power.t.test(n = n(psi)/2,
               delta = pp(psi, "muA") - pp(psi, "muB"),
               sd = theta(psi)
               )$power
}

## evaluating the power-function for all combinations of n and theta
calc <- powCalc(psi, powFun)

## adding example at theta of 1 and power of 0.9
pow <- powEx(calc, theta = 1, power = 0.9)

## drawing an inspection plot
inspect(pow)
```

`plot` *Power Plot*

Description

A sensitivity plot (called power plot) for the sample size calculation. Using a contour for a given power, it shows how sample size changes if `theta` is varied.

Usage

```
plot(x, y, ...)
```

Arguments

`x`	The object of class power used for plotting
`y`	Not used
`...`	additional arguments implemented:

- `at = c(0.9, 0.8, 0.85, 0.95)` a numeric vector giving breakpoints along the power range. Contours (if any) will be drawn at these values. The contour line of the example will be emphasised. If `example = FALSE` the first number indicates, which contour should be emphasized.

- `smooth = FALSE` logical that indicates if the contours should be smoothed. If TRUE a span of 0.75 will be used by default. Alternatively the argument smooth can also take a numeric value that will be used for smoothing.See the documentation of the function `loess` for details.
- `example = TRUE` a logical indicating if an example should be drawn or not. An example is an arrow that points from the particular `theta` on the x-axis to the contour line and to the sample size on the y-axis.
- `reflines = TRUE` a logical indicating if reference lines should be drawn or not. Reference lines are drawn at every n and theta that was used for evaluating the power function. If reference lines are drawn the background will be grey.

Details

Generates a contour plot with theta on the x-axis and n on the y-axis and the contours for the estimated power (indicated with the argument at).

Value

A plot is generated but nothing is returned.

See Also

`inspect` for drawing an inspection plot and `levelplot` for further arguments that can be passed to plot.

Examples

```
## defining the range of n and theta to be evaluated
psi <- powPar(theta = seq(from = 0.5, to = 1.5, by = 0.1),
              n = seq(from = 20, to = 60, by = 2),
              muA = 0,
              muB = 1)

## defining a power-function
powFun <- function(psi){
  power.t.test(n = n(psi)/2,
               delta = pp(psi, "muA") - pp(psi, "muB"),
               sd = theta(psi)
               )$power
}

## evaluating the power-function for all combinations of n and theta
calc <- powCalc(psi, powFun)

## adding example at theta of 1 and power of 0.9
pow <- powEx(calc, theta = 1, power = 0.9)

## drawing the power plot with 3 contour lines
plot(pow,
     xlab = "Standard Deviation",
```

```
     ylab = "Total Sample Size",
     at = c(0.85, 0.9, 0.95))

## without example the contour line at the first element of at is bold
plot(pow, example = FALSE)
```

`powCalc` *Power calculation*

Description

The user-defined 'power-function' provided as `statistic` will be evaluated for the whole range of `n`, `theta`, and `xi` as specified in the powPar-object.

Usage

```
powCalc(object, statistic, n.iter = NA, cluster = FALSE)
```

Arguments

`object`	An object of class `powPar`.
`statistic`	A function that takes an object of class `powPar` as argument. Ideally this is also the only argument. The function should return a vector of numeric values or a vector of logicals, depending on the `type`. See Details.
`n.iter`	A number specifying how often the power-function is evaluated.
`cluster`	Still experimental! This argument can be logical, indicating if the library parallel should be used or not, or numeric. In the latter case the number is passed as integer to the function makeCluster from library parallel. The default is FALSE.
`...`	Not used so far.

Details

If the statistic does not return the power (a numeric value between 0 and 1) but returns a logical (TRUE or FALSE) the argument `n.iter` is expected. The statitic will then be evaluated n.iter-times and the proportion of successes will be interpreted as the power.

Value

An object of class powCalc.

Examples

```
## defining the range of n and theta to be evaluated
psi <- powPar(theta = seq(from = 0.5, to = 1.5, by = 0.1),
              n = seq(from = 20, to = 60, by = 2),
              muA = 0,
              muB = 1)

## defining a power-function
powFun <- function(psi){
  power.t.test(n = n(psi)/2,
               delta = pp(psi, "muA") - pp(psi, "muB"),
               sd = theta(psi)
               )$power
}

## evaluating the power-function for all combinations of n and theta
calc <- powCalc(psi, powFun)
```

---

powEx *Defining the example to be used and the method to be used for sample size estimation.*

Description

A function for constructing an object of class power used for drawing an example in a sensitivity plot and for estimating the sample size.

Usage

```
powEx(x, theta, xi = NA, endpoint = NA, power = 0.9, drop = 0,
             method = c("default", "lm", "step"), lm.range = NA,
             forceDivisor = FALSE)
```

Arguments

x	An object of class powCalc.
theta	a numeric value indicating for which theta to draw the example in the sensitivity plot and where to evaluate sample size. It makes only sense to use a theta in the range evaluated.
xi	a numeric value, as theta but for xi
endpoint	Object of class character, indidating for which endpoint sample size should be evaluated
power	Object of class numeric, indicating for what power samle size should be evaluated
method	Defining the method how the sample size for the is calculated. method = "default" uses "lm" if resampling was used to calculate the powCalc object, otherwise "step" is used.

lm.range	The range of evaluations that are used for estimating the sample size if the method = "lm" or evaluates to "lm". For the default lm.range = 0.2 this means from 80 to 120 % of the power in the example, e.g. for the power of 0.9 this is a range from 0.72 to 1.08. Note that the range is cut at 0 and 1.
drop	Object of class numeric (range: 0 to 1), indicating how many drop outs are expected. This information is used to calculate the number of subject that should be recruited (addressed e.g. by the function tex using type nRec).
forceDivisor	If TRUE the biggest common divisor of all evaluated sample sizes is used as divisor and the estimated sample size is increased to be divisible by this divisor. If an integer is provided it is used as divisor.

Details

For method equal to "lm" a linear model is fit as lm(sample.size ~ transformed(power)) with all data where theta, and xi are equal to the theta and xi of the example and within the power-range as defined by the argument lm.range. This model is then used for predicting the sample size. Always inspect the result using inspect!

The method "step" returns the last element in the sequence of sample sizes - power pairs, sorted with decreasing power, where the power is above the power defined for the example.

Value

An object of class power.

Note

In older verstions of the package: The function merge was used together with an object of class powEx to form an object of class power.

Examples

```
## defining the range of n and theta to be evaluated
psi <- powPar(theta = seq(from = 0.5, to = 1.5, by = 0.1),
              n = seq(from = 20, to = 60, by = 2),
              muA = 0,
              muB = 1)

## defining a power-function
powFun <- function(psi){
  power.t.test(n = n(psi)/2,
               delta = pp(psi, "muA") - pp(psi, "muB"),
               sd = theta(psi)
               )$power
}

## evaluating the power-function for all combinations of n and theta
calc <- powCalc(psi, powFun)

## adding example at theta of 1 and power of 0.9
pow <- powEx(calc, theta = 1, power = 0.9)
```

```
## drawing the power plot with 3 contour lines
plot(pow,
     xlab = "Standard Deviation",
     ylab = "Total Sample Size",
     at = c(0.85, 0.9, 0.95))

## changing the estimation method
pow2 <- powEx(calc, theta = 1, power = 0.9, method = "lm")

## drawing an inspection plot
inspect(pow2)
```

---

powPar *Constructing an object of class 'powPar'.*

Description

A function for constructing an object of class powPar. Such an object is used for evaluating the user defined 'power function' for a parameter range. All information that is needed for calculating the power (e.g. a pilot data set) should be provided by making use of the `...` argument.

Usage

```
powPar(n, theta = NA, xi = NA, ...)
```

Arguments

`n`	A numeric vector, indicating for which sample sizes to evaluate the power function.
`theta`	A numeric vector that will be used for evaluating the power function. The method `theta` can be used within the power function to extract the elements of this vector one by one.
`xi`	A numeric vector that will be used for evaluating the power function. Since for every element of`xi` an individuall sensitivity plot has to be constructed, the length of the `xi` vector is usually short.
`...`	This arguemt is used to provide additional parameters needed by the power function for calculating the power. This parameters can be extracted using the function pp.

Details

An object of class powPar is used to evaluate the 'power function' for a range of `n` and `theta` and optionally for several `xi` values.

The user can write a 'power function' and extract the individual elements using the functions n, theta, xi and pp.

It is a good practice to include everything that is needed for the calculation, also data sets etc.

To extract the vector of `theta`, instead of individual values, you can use the method `pp` with the name theta.

For historical reasons: If the argument `theta = NA` the argument `theta.name` (a character) has to be used, to indicate the name of a numeric vector that was passed to the argument (`...`). The same is true for the argument `xi`.

Value

An object of the class powPar

Examples

```
## defining the range of n and theta to be evaluated
psi <- powPar(n = seq(from = 20, to = 60, by = 2),
              theta = seq(from = 0.5, to = 1.5, by = 0.05)
              )

## defining a power-function
powFun <- function(psi){
  return(power.t.test(n = n(psi)/2, delta = theta(psi), sig.level = 0.05)$power)
}

## evaluating the power-function for all combinations of n and theta
calc <- powCalc(psi, statistic = powFun)

## adding example at theta of 1 and power of 0.9
pow <- powEx(calc, theta = 1)

## drawing the power plot
plot(pow,
     xlab = "Difference",
     ylab = "Total Sample Size")
```

---

refine *Refining the estimation*

Description

Increasing the number of iterations for estimating the sample size for the 'theta' and 'xi' as specified for the example.

Usage

```
refine(object, factor = 10)
```

Arguments

`object`	An object of class power.
`factor`	An integer larger than one that is multiplied with the available number of iterations to from the target number of iterations.

Value

An object of class power.

Note

This function is only useful if the object of class power was generated using a resamling approach.

Examples

```
 ## takes quite some time
## defining the range of n and theta to be evaluated
psi <- powPar(theta = seq(from = 0.5, to = 1.5, by = 0.1),
              n = seq(from = 20, to = 60, by = 2))

## defining a power-function
powFun <- function(psi){
  x <- rnorm(n(psi)/2)
  y <- rnorm(n(psi)/2) + theta(psi)
  return(wilcox.test(x = x, y = y)$p.value < 0.05)
}


## evaluating the power-function for all combinations of n and theta
calc <- powCalc(psi, powFun, n.iter = 10)

## adding example at theta of 1 and power of 0.9
pow <- powEx(calc, theta = 1, power = 0.9)

## another 900 (= 1000 - 100) iterations
refine(pow)
```

---

tex *Preparing text for using with LaTeX*

Description

Methods for function tex

Usage

```
tex(x, type, ...)
```

Arguments

x	The object of class power used for extraction
type	Currently available: • "drop", indicating the drop-out rate used for calculation.

- "nRec", sample size that was increased to take into account the drop-out rate.
- "nEval", sample size needed for evaluation without taking into account the drop-out rate.
- "n.iter", nuber of iterations used for calculation.
- "power", 'power' used for calculation.
- "sampling", a description of the sampling process.
- "theta", 'theta' used for calculation.

... Not used so far

Value

A character string.

Methods

This methods prepare strings that can directly be used for including information from objects of power into Sweave reports.

Examples

```
## defining the range of n and theta to be evaluated
psi <- powPar(theta = seq(from = 0.5, to = 1.5, by = 0.1),
              n = seq(from = 20, to = 60, by = 2),
              muA = 0,
              muB = 1)

## defining a power-function
powFun <- function(psi){
  power.t.test(n = n(psi)/2,
               delta = pp(psi, "muA") - pp(psi, "muB"),
               sd = theta(psi)
               )$power
}

## evaluating the power-function for all combinations of n and theta
calc <- powCalc(psi, powFun)

## adding example at theta of 1 and power of 0.9
pow <- powEx(calc, theta = 1, power = 0.9)

## drawing the power plot with 3 contour lines
plot(pow,
     xlab = "Standard Deviation",
     ylab = "Total Sample Size",
     at = c(0.85, 0.9, 0.95))

##
tex(pow, type = "sampling")
```

update *Updating a powCalc or a power object.*

Description

A function for updating an existing object of class powCalc or power.

Usage

```
update(object, ...)
```

Arguments

object	An object of class powCalc or power.
...	The following elements (slots) of the object can be updated:

n.iter A number indicating the number of iterations used to dertermine the power if the calcualation is based on resampling. The existing iterations will be kept. n.iter indicates the number of iterations after evaluation, therefore n.iter has to be equal or larger than the existing number of iterations.

n A vector with numbers for evaluating the power. New elements will be evaluated and existing elements reused. If some elements of the original are not part of n they will be ommitted.

theta see n for details.

xi see n for details.

statistic A function of an object of class psi. If a new statistic is provided all elements will be evaluated again.

Value

An object of class powCalc.

Note

Be careful if you use this function to update objects of class power.

See Also

powCalc for geneating new objcets of class powCalc.

Examples

```
## defining the range of n and theta to be evaluated
psi <- powPar(theta = seq(from = 0.5, to = 1.5, by = 0.1),
              n = seq(from = 20, to = 60, by = 2),
              muA = 0,
              muB = 1)
```

```
## defining a power-function
powFun <- function(psi){
  power.t.test(n = n(psi)/2,
               delta = pp(psi, "muA") - pp(psi, "muB"),
               sd = theta(psi)
               )$power
}

## evaluating the power-function for all combinations of n and theta
calc <- powCalc(psi, powFun)

## updating by using additional elements for "n"
calc2 <- update(calc, n = seq(from = 20, to = 90, by = 2))

## adding example at theta of 1 and power of 0.9
pow <- powEx(calc2, theta = 1, power = 0.9)

## drawing the power plot with 3 contour lines
plot(pow,
     xlab = "Standard Deviation",
     ylab = "Total Sample Size",
     at = c(0.85, 0.9, 0.95))
```

Index