Package: equivUMP (via r-universe)

October 17, 2024

Type Package

Title Uniformly Most Powerful Invariant Tests of Equivalence

Version 0.1.1

Description Implementation of uniformly most powerful invariant equivalence tests for one- and two-sample problems (paired and unpaired) as described in Wellek (2010, ISBN:978-1-4398-0818-4). Also one-sided alternatives (non-inferiority and non-superiority tests) are supported. Basically a variant of a t-test with (relaxed) null and alternative hypotheses exchanged.

License GPL (>=2)

URL https://github.com/thmild/equivUMP

BugReports https://github.com/thmild/equivUMP/issues

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Repository https://thmild.r-universe.dev

RemoteUrl https://github.com/thmild/equivump

RemoteRef HEAD

RemoteSha 99ff7bfe45fc3decd80d5d3b26a35afe720996fc

Contents

`equiv.test` *Equivalence and non-inferiority tests for one- and two-sample problems*

Description

Implementation of uniformly most powerful invariant equivalence tests for one- and two-sample problems (paired and unpaired). Also one-sided alternatives (non-inferiority and non-superiority tests) are supported. Basically a variant of a t-test with (relaxed) null and alternative hypotheses exchanged.

Usage

```
equiv.test(x, ...)

## Default S3 method:
equiv.test(x, y = NULL, alternative = c("two.sided",
  "less", "greater"), eps = 1, mu = 0, paired = FALSE, ...)

## S3 method for class 'formula'
equiv.test(formula, data, subset, na.action, ...)
```

Arguments

`x`	a (non-empty) numeric vector of data values.
`...`	further arguments to be passed to or from methods.
`y`	an optional (non-empty) numeric vector of data values.
`alternative`	a character string specifying the alternative hypothesis, must be one of "`two.sided`" (default), "`greater`" or "`less`". You can specify just the initial letter.
`eps`	a single strictly positive number giving the equivalence limits.
`mu`	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
`paired`	a logical indicating whether you want a paired equivalence test in the two-sample case.
`formula`	a formula of the form `lhs ~ rhs` where `lhs` is a numeric variable giving the data values and `rhs` a factor with two levels giving the corresponding groups.
`data`	an optional matrix or data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used.
`na.action`	a function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.

Details

`equiv.test` is modelled after (and borrows code from) R's `t.test()` and is intended to work as similarly as possible.

This functions implements uniformly most powerful invariant equivalence tests for one-sample and (paired or unpaired) two-sample problems. Also supported are one-sided versions (so-called non-inferiority or non-superiority tests).

All tests are on standardized (differences of) means $theta$:

$$theta = (mu_x - mu)/sigma$$

for the one-sample case,

$$theta = (mu_d - mu)/sigma_d$$

for the paired two-sample case and

$$theta = (mu_x - mu_y - mu)/sigma$$

for the unpaired test, where $sigma$ is the standard deviation of x and y and $sigma_d$ is the standard deviation of the differences. mu is a shift parameter that can be used to compare against a known value in the one-sample case. mu should usually be zero for two-sample problems.

The null and alternative hypotheses in equivalence tests (`alternative = "two.sided"`) are

$$H_0 : theta <= -eps \qquad or \qquad theta >= eps$$

vs

$$H_1 : -eps < theta < eps$$

Currently, only symmetric equivalence intervals $(-eps, eps)$ are supported.

In the non-inferority-case (`alternative = "greater"`) we test

$$H_0 : theta <= -eps$$

vs

$$H_1 : theta > -eps$$

In the non-superiority-case (`alternative = "less"`) we test

$$H_0 : theta >= eps$$

vs

$$H_1 : theta < eps$$

If `paired` is `TRUE` then both `x` and `y` must be specified and they must be the same length. Missing values are silently removed (in pairs if `paired` is `TRUE`).

The formula interface is only applicable for the two-sample tests.

Value

A list with class `htest` containing the following components:

`statistic`	the value of the t-statistic.
`parameter`	the degrees of freedom for the t-statistic.
`p.value`	the p-value for the test.
`estimate`	the plug-in estimate of the standardized mean (or mean difference), i.e. the empirical mean (or difference of empirical means) divided by the empirical standard deviation. Note that this estimate is not unbiaded.
`null.value`	non-equivalence limits, i.e. boundaries of null hypothesis
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string indicating what type of equivalence test was performed.
`data.name`	a character string giving the name(s) of the data.

Methods (by class)

- `default`: Default S3 method:
- `formula`: S3 method for class 'formula'

References

Wellek, S. (2010). Testing Statistical Hypotheses of Equivalence and Noniferiority. Second edition. Boca Raton: Chapman & Hall. (especially Chapters 5.3 and 6.1).

Examples

```
# compare two feed from chickwts dataset
data("chickwts")
chickwts2 <- chickwts[chickwts$feed %in% c("linseed", "soybean"),]
chickwts2$feed <- droplevels(chickwts2$feed)

# similar but cannot be shown to be equivalent up to 0.5 sigma at 0.05 level^
plot(weight ~ feed, data = chickwts2)
equiv.test(weight ~ feed, data = chickwts2, eps = 0.5)
```

Index