

Package ‘pairwiseCI’

October 14, 2009

Type Package

Title Confidence intervals for two sample comparisons

Version 0.1-17

Date 2008-10-12

Author Frank Schaarschmidt, Daniel Gerhard

Maintainer Frank Schaarschmidt <schaarschmidt@biostat.uni-hannover.de>

Depends R (>= 2.2.0), stats, exactRankTests, boot, mratios, binMto, MASS, MCPAN

Description Calculation of the parametric, nonparametric confidence intervals for the difference or ratio of location parameters and for the difference, ratio and odds-ratio of binomial proportions for comparison of independent samples. CI are not adjusted for multiplicity. A by statement allows calculation of CI separately for the levels of further factors. Please note that, when a (generalized) linear model can be reasonably assumed, there are smarter methods for CI calculation available than are implemented in this package!

License GPL

Repository CRAN

Date/Publication 2009-10-14 13:13:22

R topics documented:

pairwiseCI-package	2
as.data.frame.pairwiseCI	4
as.data.frame.pairwiseMEP	5
bean	7
behenic	7
cabbage	8
dieldrin	9
Oats	9
pairwiseCI	10
pairwiseCIInt	16

pairwiseCImethodsCont	18
pairwiseCImethodsCount	21
pairwiseCImethodsProp	23
pairwiseMEP	28
pairwiseTest	29
pairwiseTestInt	33
plot.pairwiseCI	35
plotCI.pairwiseMEP	36
print.pairwiseCI	37
print.pairwiseTest	37
print.summary.pairwiseCI	38
print.summary.pairwiseTest	38
profileDG	39
Prop.test	39
repellent	41
rooting	42
sodium	43
summary.pairwiseCI	43
summary.pairwiseTest	44
Index	46

pairwiseCI-package *Wrapper functions for two-sample confidence intervals and tests.*

Description

A collection of wrapper functions for simple evaluation of factorial trials. The function `pairwiseCI` allows to calculate 2-sample confidence intervals for all-pairs and many-to-one comparisons between levels of a factor. Intervals are NOT adjusted for multiple hypothesis testing per default. The function `pairwiseTest` allows to calculate p-values of two-sample tests for all-pairs and many-to-one comparisons between levels of a factor. P-values are NOT adjusted for multiple hypothesis testing per default. Both function allow splitting of the data according to additional factors. Intervals can be plotted, `summary.pairwiseTest` allows to use the p-value adjustments as implemented in `p.adjust(stats)`. Different test and interval methods (parametric, nonparametric, bootstrap for robust estimators of location, binomial proportions) are implemented in a unified user level function.

Author(s)

Frank Schaarschmidt and Daniel Gerhard, for the Institute of Biostatistics, Leibniz Universitaet Hannover
 Maintainer: Frank Schaarschmidt <schaarschmidt@biostat.uni-hannover.de>

See Also

Multiple comparisons for the differences of means:**multcomp**
`pairwise.t.test(stats)` `pairwise.prop.test(stats)` `p.adjust(stats)`

Examples

```

# some examples:
# In many cases the shown examples might not make sense,
# but display how the functions can be used.

data(Oats)
Oats

# # all pairwise comparisons,
# separately for each level of nitro:

apc <- pairwiseCI(yield ~ Variety, data=Oats,
  by="nitro", method="Param.diff")

apc

# Intervals can be plotted:

plot(apc)

# See ?pairwiseCI or ?pairwiseCImethodsCont
# for further options for intervals of 2 samples
# of continuous data.

# Or a test

apcTest <- pairwiseTest(yield ~ Variety, data=Oats,
  by="nitro", method="t.test")

# with holm-adjusted p-values:
summary(apcTest, p.adjust.method="holm")

# # If only comparisons to one control would be of interest:
# many to one comparisons, with variety Marvellous as control,
# for each level of nitro separately:

m2l <- pairwiseCI(yield ~ Variety, data=Oats,
  by="nitro", method="Param.diff", control="Marvellous")

#####
# # Proportions: another structure of the data is needed.
# Currently the data.frame data must contain two columns,
# specifying the number of successes and failures on each
# unit.

# The rooting example:
# Calculate confidence intervals for the
# difference of proportions between the 3 doses of IBA,
# separately for 4 combinations of "Age" and "Position".
# Note: we pool over Rep in that way. Whether this makes

```

```

# sense or not, is decision of the user.

data(rooting)
rooting

# Confidence intervals for the risk difference

aprootsRD<-pairwiseCI(cbind(root, noroot) ~ IBA,
  data=rooting, by=c("Age", "Position"), method="Prop.diff")

# See ?pairwiseCIProp for further options to compare proportions

# Or a test:

aprootsTest<-pairwiseTest(cbind(root, noroot) ~ IBA,
  data=rooting, by=c("Age", "Position"), method="Prop.test")
aprootsTest

summary(aprootsTest, p.adjust.method="holm")

```

```

as.data.frame.pairwiseCI
      Coercing "pairwiseCI" to a "data.frame"

```

Description

Creates a data.frame from the output of pairwiseCI.

Usage

```

## S3 method for class 'pairwiseCI':
as.data.frame(x, ...)

```

Arguments

x	an object of class "pairwiseCI"
...	currently not used

Value

A data.frame with the columns

estimate	containing the estimates
lower	containing the lower bounds
upper	containing the upper bounds
comparison	containing character strings, specifying which levels have been compared
by	containing the levels by which the original data set has been split

and the `conf.level` and the a character string naming the used method as attributes `"conf.level"` and `"methodname"`.

See Also

`pairwiseCI`, `summary.pairwiseTest`

Examples

```
data(bean)

out<-pairwiseCI(yield~P, data=bean, by="T", method="Param.diff",
  var.equal=FALSE, alternative="greater")

out

dat<-as.data.frame(out)

dat

str(dat)

# # # #

data(repellent)

out2<-pairwiseCI(decrease~treatment, data=repellent, control="H",
  alternative="two.sided", method="Param.ratio")

out2

as.data.frame(out2)
```

```
as.data.frame.pairwiseMEP
  Coerce "pairwiseMEP" to "data.frame"
```

Description

Coerces the output of the function `pairwiseMEP` to a `data.frame`.

Usage

```
## S3 method for class 'pairwiseMEP':
as.data.frame(x, row.names = NULL, optional = FALSE, whichep = NULL, ...)
```

Arguments

<code>x</code>	an object of class 'pairwiseMEP' as can be obtained by calling <code>pairwiseMEP</code>
<code>row.names</code>	as in <code>as.data.frame</code>
<code>optional</code>	as in <code>as.data.frame</code>
<code>whichep</code>	a vector of integers or character strings, indexing which endpoints (which response variables) from the <code>x</code> shall be coerced to a data.frame; if omitted (default), all endpoints are coerced to a data.frame
<code>...</code>	Further arguments to be passed to <code>as.data.frame</code>

Value

A data.frame with columns

<code>estimate</code>	numeric, the point estimates
<code>lower</code>	numeric, the lower confidence limits
<code>upper</code>	numeric, the upper confidence limits
<code>comparison</code>	character, the name of the groupwise comparison
<code>by</code>	optional, character, the name of subset of the original data.frame
<code>response</code>	character, the name of the response variable
<code>method</code>	character, the name of the method used for calculation of the lower and upper limits, see <code>pairwiseMEP</code>

Examples

```
x1<-rnorm(120,20,2)
x2<-rnorm(120,100,8)
x3<-rpois(120,10)
x4<-rbinom(120,mu=10, size=10)
A<-rep(c("a1","a2","a3"), c(40,40,40))
B<-rep(rep(c("b1","b2","b3","b4"), c(10,10,10,10)), times=3)

dat<-data.frame(x1=x1, x2=x2, x3=x3, x4=x4, A=A, B=B)

test<-pairwiseMEP(x=dat,
  ep=c("x1","x2","x3","x4"),
  f="A", by="B",
  method=c("Param.ratio","Param.ratio","Negbin.ratio","Negbin.ratio"))

test

as.data.frame(test)

as.data.frame(test, welchep=c(1,2))

as.data.frame(test, welchep=c(3,4))
```

```
as.data.frame(test, whichep=c("x1", "x2"))
```

bean	<i>Yield of beans in a two factorial field trial</i>
------	--

Description

Block design with four blocks. Within the blocks a 2x3 factorial design was randomized: Two types of horse beans (*Vicia faba*) were grown under three fertilization levels of phosphor.

Usage

```
data(bean)
```

Format

A data frame with 24 observations on the following 4 variables.

block a numeric vector

T a factor with levels, describing the plant type, where T1 = short, and T2 = tall

P a factor with levels, the phosphorus rate with levels P0 = 0, P25 = 25 kg/ha P50 = 50 kg/ha

yield a numeric vector, giving the mean yields of beans in kg/ha

Source

Petersen, RG: Design and Analysis of experiments. Marcel Dekker, Inc.

Examples

```
data(bean)
boxplot(yield ~ P*T, data=bean)
```

behenic	<i>Measurements of behenic acid in two samples</i>
---------	--

Description

Observations below a detection limit: in field trials with transgenic and isogenic corn varieties the behenic acid content was measured. Objective is to prove equivalence.

Usage

```
data(behenic)
```

Format

A data frame with 12 observations on the following 2 variables.

Treatment a factor with 2 levels: `transgenic` `xisogenic`

Behenic a numeric vector giving concentration of behenic acid, where 0.002 is the detection limit

Source

Oberdoerfer, R.B. Example dataset from composition analyses of genetically modified oilseed rape seeds. 2003; BCS GmbH.

Examples

```
data (behenic)
boxplot (Behenic~Treatment, data= behenic)
```

cabbage

Yield of cabbage in one-factorial field trial

Description

Cabbage yield in dependence of four doses of fertilizer.

Usage

```
data (cabbage)
```

Format

A data frame with 20 observations on the following 2 variables.

Fert a factor with levels `D1` `D2` `D3` `D4`, the different doses of fertilizer

yield a numeric vector, cabbage yield

Examples

```
data (cabbage)
boxplot (yield~Fert, data=cabbage)
```

`dieldrin`*Dieldrin in fish from two different rivers*

Description

Observations below a detection limit: the dieldrin concentration (microg/kg) in roach fish (*Rutilus rutilus*) were measured in the Thames river near Oxford, UK and as a control in an upper arm, the river Ray.

Usage

```
data(dieldrin)
```

Format

A data frame with 11 observations on the following 2 variables.

River a factor with 2 levels aThames Ray

dieldrin a numeric vector giving the measured dieldrin concentration, where 0.09 was the detection limit

Source

Yamaguchi N. et al. Concentrations and hazard assessment of PCBs, organochlorine pesticides and mercury in fish species from the upper Thames: River pollution and its potential effects on top predators. *Chemosphere* 2003, 50, 265-273.

Examples

```
data(dieldrin)
boxplot(dieldrin~River, data=dieldrin)
```

`Oats`*The Oats data set*

Description

The yield of three varieties of Oat was recorded in a field trial with 6 Blocks on 4 levels of nitrogen fertilization. Originally a split plot design, here simply for demonstration of pairwiseCI.

Usage

```
data(Oats)
```

Format

A data frame with 72 observations on the following 4 variables.

Block an ordered factor with levels VI < V < III < IV < II < I

Variety a factor with levels Golden Rain Marvellous Victory

nitro a numeric vector

yield a numeric vector

Source

See Oats(nlme).

Examples

```
data(Oats)
boxplot(yield ~ nitro*Variety, data=Oats)
```

pairwiseCI

Wrapper function for two-sample confidence intervals

Description

Confidence intervals (CI) for difference or ratio of location parameters of two independent samples. The CI are NOT adjusted for multiplicity by default. A `by` statement allows for separate calculation of pairwise comparisons according to further factors in the given dataframe. The function applies the intervals available from `t.test(stats)` for difference of means with and without assumptions of homogeneous variances; large sample approximations for the difference and ratio of means of lognormal data; the exact CI for difference (`wilcox.exact(exactRankTests)`) and ratio of location based on the Hodges-Lehmann estimator; bootstrap intervals for ratio and difference of Medians and Harrell-Davis estimators a more robust alternatives to the Hodges-Lehmann estimator (`boot`, `Hmisc`); the Score test derived CI for the difference (`prop.test(stats)`), Woolf-interval for the odds-ratio and a crude asymptotic as well as an iterative interval for the ratio of proportions.

Usage

```
pairwiseCI(formula, data, by = NULL,
            alternative = "two.sided", conf.level = 0.95,
            method = "Param.diff", control = NULL, ...)
```

Arguments

<code>formula</code>	A formula of the structure <code>response ~ treatment</code> for numerical variables, and of structure <code>cbind(success, failure) ~ treatment</code> for binomial variables
<code>data</code>	A <code>data.frame</code> containing the numerical response variable and the treatment and <code>by</code> variable as factors. Note, that for binomial data, two columns containing the number of successes and failures must be present in the data.

by	A character string or character vector giving the name(s) of additional factors by which the data set is to be split.
alternative	Character string, either "two.sided", "less" or "greater"
conf.level	The comparisonwise confidence level of the intervals, where 0.95 is default
method	A character string specifying the confidence interval method, with the following options <i>"Param.diff"</i> : Difference of two means, with additional argument <i>var.equal=FALSE</i> (default) as in <i>t.test(stats)</i> , <i>"Param.ratio"</i> : Ratio of two means, with additional argument <i>var.equal=FALSE</i> (default) as in <i>t.test.ratio(mratios)</i> , <i>"Lognorm.diff"</i> : Difference of two means, assuming a lognormal distribution, based on generalized pivotal quantities, <i>"Lognorm.ratio"</i> : Ratio of two means, assuming a lognormal distribution, based on generalized pivotal quantities, <i>"HL.diff"</i> : Exact conditional nonparametric CI for difference of locations based on the Hodges-Lehmann estimator, <i>"HL.ratio"</i> : Exact conditional nonparametric CI for ratio of locations, based on the Hodges-Lehmann estimator, <i>"HD.diff"</i> : Nonparametric CI for difference of locations, based on the Harrell-Davis estimator (percentile bootstrap, stratified by the grouping variable), <i>"HD.ratio"</i> : Nonparametric CI for ratio of locations, based on the Harrell-Davis estimator (percentile bootstrap, stratified by the grouping variable), <i>"Median.diff"</i> : Nonparametric CI for difference of locations, based on the medians (percentile bootstrap, stratified by the grouping variable), <i>"Median.ratio"</i> : Nonparametric CI for ratio of locations, based on the medians (percentile bootstrap, stratified by the grouping variable), <i>"Prop.diff"</i> : Asymptotic continuity corrected CI for difference of proportions as implemented in <i>prop.test(stats)</i> , <i>"Negbin.ratio"</i> : Profile-likelihood CI for ratio of expected values assuming the negative binomial distribution, adapting code from the profile function of MASS (Venables and Ripley,2002), <i>"Quasipoisson.ratio"</i> : Profile-deviance CI for the ratio of expected values with a quasipoisson assumption, using the adapted code of the profile function in MASS <i>"Poisson.ratio"</i> : Profile-likelihood CI for the ratio of expected values with a Poisson assumption, again using slightly changed code from MAass, <i>"Prop.diffAdd2"</i> : Approximate CI for difference of proportions according to Agresti and Caffo (2000), <i>"Prop.ratio"</i> : Asymptotic CI (normal approximation on the log: <i>CImethod="GNC"</i>) or iterative CI (<i>CImethod="Score"</i>) for ratio of proportions, <i>"Prop.or"</i> : Asymptotic CI for the odds ratio (normal approximation on the log: <i>CImethod="Wolf"</i>), or the exact CI corresponding to Fishers exact test (<i>CImethod="Exact"</i> , taken from <i>fisher.test</i> , package <i>stats</i>) See <i>?pairwiseCImethods</i> for details.
control	Character string, specifying one of the levels of the treatment variable as control group in the comparisons; default is NULL, then CI for all pairwise comparisons are calculated.
...	further arguments to be passed to the functions specified in pairwiseCImethodsCont , and pairwiseCImethodsProp

Details

Note that all the computed intervals are without adjustment for multiplicity by default. When based on crude normal approximations or crude non-parametric bootstrap methods, the derived confidence

intervals can be unreliable for small sample sizes. The method implemented here split the data set into small subsets (according to the grouping variable in *formula* and the variable specified in *by*) and compute confidence intervals only based each particular subset. Please note that, when a (generalized) linear model can be reasonable assumed to describe the complete data set, it is smarter to compute confidence intervals based on the model estimators. Methods to do this are, e.g., implemented in the packages **stats**, **multcomp**, **mratios**.

Value

an object of class "pairwiseCI" structured as:

a list containing:

byout	A named list, ordered by the levels of the by-factor, where each element is a list containing the numeric vectors <i>estimate</i> , <i>lower</i> , <i>upper</i> and the character vector <i>compnames</i>
byname	the level names of the by-factor

and further elements as input, try `str()` for details.

the object is printed by `print.pairwiseCI`.

References

- *Param.diff* simply uses: `t.test` in **stats**, note that the default setting assumes possibly heterogeneous variances (`var.equal=FALSE`).
- *Param.ratio* for homogeneous variances (`var.equal=TRUE`): **Fieller EC (1954)**: Some problems in interval estimation. *Journal of the Royal Statistical Society, Series B*, 16, 175-185.
- *Param.ratio* for heterogenous variances (`var.equal=FALSE`): : the test proposed in: **Tamhane, AC, Logan, BR (2004)**: Finding the maximum safe dose level for heteroscedastic data. *Journal of Biopharmaceutical Statistics* 14, 843-856. is inverted by solving a quadratic equation according to Fieller, where the estimated ratio is simply plugged in order to get Satterthwaite approximated degrees of freedom. See also: **Hasler, M, Vonk, R, Hothorn, LA**: Assessing non-inferiority of a new treatment in a three arm trial in the presence of heteroscedasticity. *Statistics in Medicine* 2007 (in press).
- *Lognorm.ratio* and *Lognorm.diff*: **Chen, Y-H, Zhou, X-H (2006)**: Interval estimates for the ratio and the difference of two lognormal means. *Statistics in Medicine* 25, 4099-4113.
- *HL.diff*: `wilcox.exact` in **exactRankTests**
- *HL.ratio*: **Hothorn, T, Munzel, U**: Non-parametric confidence interval for the ratio. Report University of Erlangen, Department Medical Statistics 2002; available via: <http://www.imbe.med.uni-erlangen.de/~hothorn/> For the Hodges-Lehmann estimator see: **Hodges, J.L., Lehmann E.L.(1963)**: Estimates of location based on rank tests. *Ann. Math. Statist.* 34, 598-611.
- *HD.diff*: The interval is calculated from a bootstrap sample (stratified according to the group variable) of the difference of Harrel-Davis estimators, using package `boot`.
- *HD.ratio*: The interval is calculated from a bootstrap sample (stratified according to the group variable) of the ratio of Harrel-Davis estimators, using package `boot`. For the Harrell-Davis estimator see: **Harrell, F.E.; Davis, C.E. (1982)**: A new distribution-free quantile estimator. *Biometrika* 69, 635-640.

- *Median.diff*: The interval is calculated from a bootstrap sample (stratified according to the group variable) of the difference of sample Medians, using package boot.
- *Median.ratio*: The interval is calculated from a bootstrap sample (stratified according to the group variable) of the ratio of sample Medians, using package boot.
Note, that the 4 bootstrap versions will hardly make sense for small samples, because of the discreteness of the resulting bootstrap sample.
- *Poisson.ratio*: Profile-likelihood CI, based on the assumption of a Poisson distribution, basic method described in **Venables, W.N., Ripley, B.D. (2002)**: Modern Applied Statistics with S. Springer Verlag, New York.
- *Quasipoisson.ratio*: Profile-deviance CI, based on the quasipoisson assumption, basic method described in Venables, **W.N., Ripley, B.D. (2002)**: Modern Applied Statistics with S. Springer Verlag, New York.
- *Negbin.ratio*: Profile-likelihood CI, based on the assumption of the negative binomial distribution, basic method described in **Venables, W.N., Ripley, B.D. (2002)**: Modern Applied Statistics with S. Springer Verlag, New York.
- *Prop.diff*: simply uses *prop.test* in **stats** which currently implements the continuity corrected interval as described under the acronym CC in **Newcombe R.G. (1998)**: Interval Estimation for the Difference Between Independent Proportions: Comparison of Eleven Methods. *Statistics in Medicine* 17, 873-890.
- *Prop.diffAdd2*: calls to *Add4* in **binMto** which implements the interval adding 2 successes and 2 failures as proposed by: **Agresti, A. and Caffo, B. (2000)**: Simple and effective confidence intervals for proportions and differences of proportions result from adding two successes and two failures. *American Statistician* 54 (4), 280-288.
- *Prop.ratio* calculates the crude asymptotic interval (normal approximation on the log-scale, *CImethod="GNC"*) or the Score interval (*CImethod="Score"*), both described in: **Gart, JJ and Nam, J (1988)**: Approximate interval estimation of the ratio of binomial parameters: A review and corrections for skewness. *Biometrics* 44, 323-338.
- *Prop.or* Adjusted Woolf interval (*CImethod="Woolf"*), e.g. in: **Lawson, R (2005)**: Small sample confidence intervals for the odds ratio. *Communication in Statistics Simulation and Computation*, 33, 1095-1113. or the exact interval corresponding to Fishers exact test (*CImethod="Exact"*, taken from *fisher.test(stats)*, see references there)

See Also

`as.data.frame.pairwiseCI` to create a data.frame from the output, `summary.pairwiseCI` to create a more easily accessible list from the output, `plot.pairwiseCI` to plot the intervals. Further, see **multcomp** for simultaneous intervals for difference for various contrasts, **mratios** for simultaneous intervals for the ratio of means, **stats** *p.adjust*, *pairwise.t.test*, *pairwise.prop.test*, *pairwise.wilcox.test*, *TukeyHSD* for methods of multiple comparisons.

Examples

```
# some examples:
# In many cases the shown examples might not make sense,
# but display how the functions can be used.
```

```

data(Oats)

# # all pairwise comparisons,
# # separately for each level of nitro:

apc <- pairwiseCI(yield ~ Variety, data=Oats,
  by="nitro", method="Param.diff")

apc

plot(apc)

# # many to one comparisons, with variety Marvellous as control,
# # for each level of nitro separately:

m21 <- pairwiseCI(yield ~ Variety, data=Oats,
  by="nitro", method="Param.diff", control="Marvellous")

plot(m21)

# # the same using confidence intervals for the ratio of means:

m21 <- pairwiseCI(yield ~ Variety, data=Oats,
  by="nitro", method="Param.diff", control="Marvellous")

plot(m21, CIvert=TRUE, H0line=0.9)

#####

# The repellent data set (a trial on repellent
# effect of sulphur on honey bees): Measured was
# the decrease of sugar solutions (the higher the decrease,
# the higher the feeding, and the less the repellent effect).
# Homogeneity of variances is questionable. Which of the doses
# leads to decrease of the variable decrease compared to the
# control group "H"?

data(repellent)
boxplot(decrease ~ treatment, data=repellent)

# as difference to control (corresponding to Welch tests)
beeCIId<-pairwiseCI(decrease ~ treatment, data=repellent,
  method="Param.diff", control="H", alternative="less",
  var.equal=FALSE)
beeCIId
plot(beeCIId)

# as ratio to control:

beeCIr<-pairwiseCI(decrease ~ treatment, data=repellent,
  method="Param.ratio", control="H", alternative="less",
  var.equal=FALSE)

```

```

beeCIr
plot(beeCIr)

# Bonferroni-adjustment can be applied:

beeCIrBonf<-pairwiseCI(decrease ~ treatment, data=repellent,
  method="Param.ratio", control="H", alternative="less",
  var.equal=FALSE, conf.level=1-0.05/7)
beeCIrBonf
plot(beeCIrBonf)

#####

# Proportions:

# The rooting example:
# Calculate confidence intervals for the
# difference of proportions between the 3 doses of IBA,
# separately for 4 combinations of "Age" and "Position".
# Note: we pool over Rep in that way. Whether this makes
# sense or not, is decision of the user.

data(rooting)

# Risk difference

aprootsRD<-pairwiseCI(cbind(root, noroot) ~ IBA,
  data=rooting, by=c("Age", "Position"), method="Prop.diff")

aprootsRD

# Odds ratio

aprootsOR<-pairwiseCI(cbind(root, noroot) ~ IBA,
  data=rooting, by=c("Age", "Position"), method="Prop.or")

aprootsOR

# Risk ratio

aprootsRR<-pairwiseCI(cbind(root, noroot) ~ IBA,
  data=rooting, by=c("Age", "Position"), method="Prop.ratio")

aprootsRR

# CI can be plotted:

plot(aprootsRR)

#####

# Other available methods:

```

```

data(dieldrin)

boxplot(dieldrin~River, data=dieldrin)

pairwiseCI(dieldrin~River, data=dieldrin, method="HD.diff")
pairwiseCI(dieldrin~River, data=dieldrin, method="HD.ratio")

pairwiseCI(dieldrin~River, data=dieldrin, method="HL.diff")
pairwiseCI(dieldrin~River, data=dieldrin, method="HL.ratio")

pairwiseCI(dieldrin~River, data=dieldrin, method="Median.diff")
pairwiseCI(dieldrin~River, data=dieldrin, method="Median.ratio")

data(sodium)

pairwiseCI(dieldrin~River, data=dieldrin, method="HD.diff")
pairwiseCI(dieldrin~River, data=dieldrin, method="HD.ratio")

# CIs assuming lognormal distribution of the response:

resp<-rlnorm(n=20, meanlog = 0, sdlog = 1)
treat<-as.factor(rep(c("A","B")))
datln<-data.frame(resp=resp, treat=treat)

pairwiseCI(resp~treat, data=datln, method="Lognorm.diff")
pairwiseCI(resp~treat, data=datln, method="Lognorm.ratio")

```

pairwiseCIInt

Internal functions for pairwiseCI

Description

For internal use. Two different methods for data representable as a two numeric vectors (`pairwiseCICont`) and data representable as matrix with two columns like `cbind(successes, failures)`. Functions that split up a `data.frame` according to one factor, and perform all pairwise comparisons and comparisons to control among the levels of the factor by calling methods documented in [pairwiseCImethodsCont](#) and [pairwiseCImethodsProp](#).

Usage

```

pairwiseCICont(formula, data, alternative="two.sided",
  conf.level=0.95, method, control=NULL, ...)

pairwiseCIProp(formula, data, alternative="two.sided",
  conf.level=0.95, control=NULL, method, ...)

```

Arguments

formula	A formula of the structure <i>response ~ treatment</i> for numerical variables, and of structure <i>cbind(success, failure) ~ treatment</i> for binomial variables
data	A data.frame containing the numerical response variable and the treatment and by variable as factors. Note, that for binomial data, two columns containing the number of successes and failures must be present in the data.
alternative	Character string, either <i>"two.sided"</i> , <i>"less"</i> or <i>"greater"</i>
conf.level	The <i>comparisonwise confidence level</i> of the intervals, where 0.95 is default
method	A character string specifying the confidence interval method, one of the following options <i>"Param.diff"</i> : Difference of two means, with additional argument <i>var.equal=FALSE</i> (default) as in <i>t.test(stats)</i> <i>"Param.ratio"</i> : Ratio of two means, with additional argument <i>var.equal=FALSE</i> (default) as in <i>t.test.ratio(mratios)</i> <i>"Lognorm.diff"</i> : Difference of two means, assuming a lognormal distribution, <i>"Lognorm.ratio"</i> : Ratio of two means, assuming a lognormal distribution, <i>"HL.diff"</i> : Exact nonparametric CI for difference of locations based on the Hodges-Lehmann estimator, <i>"HL.ratio"</i> : Exact nonparametric CI for ratio of locations, based on the Hodges-Lehmann estimator, <i>"HD.diff"</i> : Nonparametric CI for difference of locations, based on the Harrell-Davis estimator (percentile bootstrap CI), <i>"HD.ratio"</i> : Nonparametric CI for ratio of locations, based on the Harrell-Davis estimator (percentile bootstrap CI), <i>"Median.diff"</i> : Nonparametric CI for difference of locations, based on the medians (percentile bootstrap CI), <i>"Median.ratio"</i> : Nonparametric CI for ratio of locations, based on the medians (percentile bootstrap CI), <i>"Prop.diff"</i> : Asymptotic CI for difference of proportions <i>prop.test(stats)</i> <i>"Prop.ratio"</i> : Asymptotic CI for ratio of proportions <i>"Prop.or"</i> : Asymptotic CI for the odds ratio See <code>?pairwiseCImethods</code> for details.
control	Character string, specifying one of the levels of the treatment variable as control group in the comparisons; default is <i>NULL</i> , then CI for all pairwise comparisons are calculated.
...	further arguments to be passed to the functions specified in <i>methods</i>

Details

These functions are for internal use in pairwiseCI.

Value

a list containing:

estimate	numeric vector: the point estimates
lower	numeric vector: lower confidence bounds
upper	numeric vector: upper confidence bounds
compnames	character vector with the names of comparisons

See Also

`pairwiseCI` for the user level function; `pairwiseCImethodsCont`, and `pairwiseCImethodsProp` for a more detailed documentation of the implemented methods; `summary.pairwiseCI` for a summary function.

`t.test(stats)`, `wilcox.exact(exactRankTests)`, `prop.test(stats)` for the sources of some of the CI methods, **multcomp** for simultaneous intervals for difference for various contrasts, **mratio**s for simultaneous intervals for the ratio in many-to-one comparisons

pairwiseCImethodsCont

Confidence intervals for two sample comparisons of continuous data

Description

Confidence interval methods available for `pairwiseCI` for comparison of two independent samples. Methods for continuous variables.

Usage

```
Param.diff(x, y, conf.level=0.95, alternative="two.sided", ...)
Param.ratio(x, y, conf.level=0.95, alternative="two.sided", ...)

Lognorm.diff(x, y, conf.level=0.95, alternative="two.sided", sim=10000, ...)
Lognorm.ratio(x, y, conf.level=0.95, alternative="two.sided", sim=10000, ...)

HL.diff(x, y, conf.level=0.95, alternative="two.sided", ...)
HL.ratio(x, y, conf.level=0.95, alternative="two.sided", ...)

Median.diff(x, y, conf.level=0.95, alternative="two.sided", ...)
Median.ratio(x, y, conf.level=0.95, alternative="two.sided", ...)

HD.diff(x, y, conf.level=0.95, alternative="two.sided", ...)
HD.ratio(x, y, conf.level=0.95, alternative="two.sided", ...)
```

Arguments

<code>x</code>	vector of observations in the first sample
<code>y</code>	vector of observations in the second sample
<code>alternative</code>	character string, either "two.sided", "less" or "greater"
<code>conf.level</code>	the comparisonwise confidence level of the intervals, where 0.95 is default
<code>sim</code>	a single integer value, specifying the number of samples to be drawn for calculation of the empirical distribution of the generalized pivotal quantities
<code>...</code>	further arguments to be passed to the individual methods, see details

Details

- *Param.diff* calculates the confidence interval for the difference in means of two Gaussian samples by calling *t.test* in package **stats**, assuming homogeneous variances if *var.equal=TRUE*, and heterogeneous variances if *var.equal=FALSE* (default);
- *Param.ratio* calculates the Fiellers (1954) confidence interval for the ratio of two Gaussian samples by calling *ratio.t.test* in package **mratios**, assuming homogeneous variances if *var.equal=TRUE*. If heterogeneous variances are assumed (setting *var.equal=FALSE*, the default), the test by Tamhane and Logan (2004) is inverted by solving a quadratic equation according to Fieller, where the estimated ratio is simply plugged in order to get Satterthwaite approximated degrees of freedom. See Hasler and Vonk (2006) for some simulation results.
- *Lognorm.diff* calculates the confidence interval for the difference in means of two Lognormal samples, based on general pivotal quantities (Chen and Zhou, 2005); currently, further arguments (...) are not used;
- *Lognorm.ratio* calculates the confidence interval for the ratio in means of two Lognormal samples, based on general pivotal quantities (Chen and Zhou, 2005); currently, further arguments (...) are not used;
- *HL.diff* calculates the Hodges-Lehmann confidence interval for the difference of locations by calling *wilcox.exact* in package **exactRankTests** ;
- *HL.ratio* calculates the Hodges-Lehmann-like confidence interval for the ratio of locations by calling *wilcox.exact* in package **exactRankTests** for the logarithms of observations;
- *HD.diff* calculates a percentile bootstrap confidence interval for the difference of “Harrell-Davis” estimates for location using code copied from *hdquantile* in package **Hmisc** and *boot.ci* in **boot**, the number of bootstrap replications can be set via *R=999* (default) ;
- *HD.ratio* calculates a percentile bootstrap confidence interval for the ratio of “Harrell-Davis” estimates for location using code copied from *hdquantile* in package **Hmisc** and *boot.ci* in package **boot**, the number of bootstrap replications can be set via *R=999* (default);
- *Median.diff* calculates a percentile bootstrap confidence interval for the difference of Medians using *boot.ci* in package **boot**, the number of bootstrap replications can be set via *R=999* (default);
- *Median.ratio* calculates a percentile bootstrap confidence interval for the ratio of Medians using *boot.ci* in package **boot**, the number of bootstrap replications can be set via *R=999* (default);

Value

A list containing:

<code>conf.int</code>	a vector containing the lower and upper confidence limit
<code>estimate</code>	a single named value

References

- *Param.diff* uses *t.test* in **stats**.
- **Fieller EC (1954)**: Some problems in interval estimation. Journal of the Royal Statistical Society, Series B, 16, 175-185.

- **Tamhane, AC, Logan, BR (2004)**: Finding the maximum safe dose level for heteroscedastic data. Journal of Biopharmaceutical Statistics 14, 843-856.
- **Hasler, M, Vonk, R, Hothorn, LA**: Assessing non-inferiority of a new treatment in a three arm trial in the presence of heteroscedasticity (submitted).
- **Chen, Y-H, Zhou, X-H (2006)**: Interval estimates for the ratio and the difference of two lognormal means. Statistics in Medicine 25, 4099-4113.
- **Hothorn, T, Munzel, U**: Non-parametric confidence interval for the ratio. Report University of Erlangen, Department Medical Statistics 2002; available via: <http://www.imbe.med.uni-erlangen.de/~hothorn/>.
- *HD.diff xxx*
- *HD.ratio xxx*
- *Median.diff xxx*
- *Median.ratio xxx*

Examples

```
#####
# Dieldrin example: Two-sample situation:
# The dieldrin example
data(dieldrin)

Ray<-subset(dieldrin, River=="Ray")$dieldrin
Thames<-subset(dieldrin, River=="aThames")$dieldrin

Ray
Thames

## CI for the difference of means,
# assuming normal errors and homogeneous variances :

thomo<-Param.diff(x=Thames, y=Ray, var.equal=TRUE)

# allowing heterogeneous variances
thetero<-Param.diff(x=Thames, y=Ray, var.equal=FALSE)

## Fieller CIs for the ratio of means,
# also assuming normal errors:

Fielhomo<-Param.ratio(x=Thames, y=Ray, var.equal=TRUE)

# allowing heterogeneous variances

Fielhetero<-Param.ratio(x=Thames, y=Ray, var.equal=FALSE)

## Hodges-Lehmann Intervalls for difference and ratios:
```

```
HLD<-HL.diff(x=Thames, y=Ray)

# allowing heterogeneous variances

HLR<-HL.ratio(x=Thames, y=Ray)

## Percentile Bootstrap intervals of Harrell-Davis estimators:

HDD<-HD.diff(x=Thames, y=Ray)

# allowing heterogeneous variances

HDR<-HD.ratio(x=Thames, y=Ray)

## Percentile Bootstrap intervals of Medians:

MedianD<-Median.diff(x=Thames, y=Ray)

# allowing heterogeneous variances

MedianR<-Median.ratio(x=Thames, y=Ray)

thomo
thetero

Fielhomo
Fielhetero

HLD
HLR

HDD
HDR

MedianD
MedianR

# # #

# Lognormal CIs:

x<-rlnorm(n=10, meanlog=0, sdlog=1)
y<-rlnorm(n=10, meanlog=0, sdlog=1)

Lognorm.diff(x=x, y=y)
Lognorm.ratio(x=x, y=y)
```

pairwiseCImethodsCount

Confidence intervals for two sample comparisons of count data

Description

Confidence interval methods available for pairwiseCI for comparison of two independent samples. Methods for count data.

Usage

```
Poisson.ratio(x, y, conf.level=0.95, alternative="two.sided")
Quasipoisson.ratio(x, y, conf.level=0.95, alternative="two.sided")
Negbin.ratio(x, y, conf.level=0.95, alternative="two.sided")
```

Arguments

x	vector of observations in the first sample
y	vector of observations in the second sample
alternative	character string, either "two.sided", "less" or "greater"
conf.level	the comparisonwise confidence level of the intervals, where 0.95 is default

Details

- *Poisson.ratio* calculates a confidence interval for the ratio of means assuming the Poisson distribution of the response by fitting a generalized linear model with log-link using *glm* in package **stats**, constructing a likelihood profile and deriving a equal-tailed confidence interval from this profile. Please note that confidence intervals from this method produce severely misleading results, when there is extra-Poisson variation in the data.
- *Quasipoisson.ratio* calculates a confidence interval for the ratio of means of the response by fitting a generalized linear model with family *quasipoisson* and log-link using *glm* in package **stats**, constructing a deviance profile and deriving a equal-tailed confidence interval from this profile.
- *Negbin.ratio* calculates a confidence interval for the ratio of means assuming the negative binomial distribution of the response by fitting a generalized linear model with log-link using *glm.nb* in package **MASS**, constructing a likelihood profile and deriving a equal-tailed confidence interval from this profile.

Note, that for all the methods, a separate glm is fitted for each two-sample comparison! When a common model can be reasonably assumed for all the data, there are smarter methods of constructing confidence intervals for groupwise comparisons, based on a common model, see e.g. the function *confint* in package **stats**, the function *confint.glm* in package **MASS** and the function *confint.glht* in package **multcomp**.

Note, that the code used here is slightly changed from the original code by Venables and Ripley, or Bates and Watts. An limit is imposed on the parameter space in which the profile is constructed.

By that limitation, intervals can also be constructed for extreme cases with all observations in one group being zero.

Note, that the *Poisson.ratio* can be used when only one count is present in each group. For *Quasipoisson.ratio*, *Negbin.ratio*, repeated observations are necessary in each group.

Value

A list containing:

<code>conf.int</code>	a vector containing the lower and upper confidence limit
<code>estimate</code>	a single named value

Author(s)

Daniel Gerhard, Frank Schaarschmidt

References

Venables WN and Ripley BD (2002). Modern Applied Statistics using S, Fourth Edition. Springer New York. **Bates, D.M. and Watts, D.G.(1988)**. Nonlinear Regression Analysis and Its Applications. John Wiley and Sons, New York.

Examples

```
library(mratios)

data(Mutagenicity)

QPCI<-pairwiseCI(MN ~ Treatment, data=Mutagenicity,
  alternative="greater", control="Vehicle", method="Quasipoisson.ratio")

plot(QPCI)
```

pairwiseCImethodsProp

Confidence intervals for two sample comparisons of binomial proportions

Description

For the comparison of two independent samples of binomial observations, confidence intervals for the difference (RD), ratio (RR) and odds ratio (OR) of proportions are implemented.

Usage

```
Prop.diff(x, y, conf.level=0.95, alternative="two.sided", CImethod=c("NHS", "CC", "GNC"))
Prop.ratio(x, y, conf.level=0.95, alternative="two.sided", CImethod=c("Score", "GNC"))
Prop.or(x, y, conf.level=0.95, alternative="two.sided", CImethod=c("Exact", "Woolf"))
```

Arguments

<code>x</code>	observations of the first sample: either a vector with number of success and failure, or a <code>data.frame</code> with two columns (the success and failures))
<code>y</code>	observations of the second sample: either a vector with number of success and failure, or a <code>data.frame</code> with two columns (the success and failures))
<code>alternative</code>	character string, either "two.sided", "less" or "greater"
<code>conf.level</code>	the comparisonwise confidence level of the intervals, where 0.95 is default
<code>CImethod</code>	a single character string, see below for details
<code>...</code>	further arguments to be passed to the individual methods, see details

Details

Generally, the input are two vectors x and y giving the number of successes and failures in the two samples, or, alternatively, two `data.frames` x and y each containing one column for the successes and one column for the failures, and the rows containing repeated observations from the same treatment. Please note, that except for function `prop.or` with `CImethod="Quasibinomial"` the confidence intervals available in this function are based on sums over the rows of x and y and hence do NOT APPROPRIATELY account for extra-binomial variability between repeated observations for the same treatment!

- `Prop.diff` calculates the asymptotic Continuity Corrected confidence interval for the difference of proportions by calling `prop.test` in package `stats`. See `?prop.test` for details. NOTE: When there are repeated observations (input as a `data.frame` with several rows), intervals are calculated based on the sums over the rows!
- `Prop.diff` with `CImethod="AC"` Calculates the Agresti-Caffo-Interval (Agresti and Caffo, 2000). NOTE: When there are repeated observations (input as a `data.frame` with several rows), intervals are calculated based on the sums over the rows!
- `Prop.diff` with `CImethod="NHS"` Calculates Newcombes Hybrid Score Interval (Newcombe, 1998). NOTE: When there are repeated observations (input as a `data.frame` with several rows), intervals are calculated based on the sums over the rows!
- `Prop.ratio` with `CImethod="GNC"` calculates the crude interval for the ratio of proportions according to Gart and Nam (1988), based on normal approximation on the log-scale. NOTE: When there are repeated observations (input as a `data.frame` with several rows), intervals are calculated based on the sums over the rows!
- `Prop.ratio` with `CImethod="Score"` calculates the Score interval for the ratio of proportions according to Gart and Nam (1988), based on a Chi-Square approximation. NOTE: When there are repeated observations (input as a `data.frame` with several rows), intervals are calculated based on the sums over the rows!

- *Prop.or* with *CImethod="Woolf"* calculates the adjusted Woolf confidence interval for the odds ratio of proportions as, e.g., described in Lawson (2005). NOTE: When there are repeated observations (input as a data.frame with several rows), intervals are calculated based on the sums over the rows!
- *Prop.or* with *CImethod="Exact"* calculates the exact confidence interval for the odds ratio of proportions corresponding to Fishers exact test, by calling to *fisher.test* in **stats**. For details, see *?fisher.test*. NOTE: When there are repeated observations (input as a data.frame with several rows), intervals are calculated based on the sums over the rows!

Value

A list containing:

<code>conf.int</code>	a vector containing the lower and upper confidence limit
<code>estimate</code>	a single named value

References

- **Newcombe R.G. (1998):** Interval Estimation for the Difference Between Independent Proportions: Comparison of Eleven Methods. *Statistics in Medicine* 17, 873-890.
- **Agresti, A. and Caffo, B. (2000):** Simple and effective confidence intervals for proportions and differences of proportions result from adding two successes and two failures. *American Statistician* 54 (4), 280-288.
- **Gart, JJ and Nam, J (1988):** Approximate interval estimation of the ratio of binomial parameters: A review and corrections for skewness. *Biometrics* 44, 323-338.
- **Dann, RS and Koch, GG (2005):** Review and evaluation of methods for computing confidence intervals for the ratio of two proportions and considerations for non-inferiority clinical trials. *Journal of Biopharmaceutical Statistics*, 15, 85-107.
- **Lawson, R (2005):** Small sample confidence intervals for the odds ratio. *Communication in Statistics Simulation and Computation*, 33, 1095-1113.
- **Venables WN and Ripley BD (2002). Modern Applied Statistics with S. Fourth Edition. Springer New York.**

Examples

```
# The rooting data.

data(rooting)

# the first comparison should be the same as:

Age5_PosB_IBA0 <- subset(rooting,
  Age=="5" & Position=="B" & IBA=="0")[,c("root", "noroot")]
Age5_PosB_IBA0.5 <- subset(rooting,
  Age=="5" & Position=="B" & IBA=="0.5")[,c("root", "noroot")]

Age5_PosB_IBA0
Age5_PosB_IBA0.5
```

```

Prop.diff(x=Age5_PosB_IBA0, y=Age5_PosB_IBA0.5)

Prop.ratio(x=Age5_PosB_IBA0, y=Age5_PosB_IBA0.5)

Prop.or(x=Age5_PosB_IBA0, y=Age5_PosB_IBA0.5)

# is the same as input two vectors x,y each containing
# the count of successes and the count of failures

colSums(Age5_PosB_IBA0)
colSums(Age5_PosB_IBA0.5)

Prop.diff(x=c(16,32),y=c(29,19))

Prop.ratio(x=c(16,32),y=c(29,19))

Prop.or(x=c(16,32),y=c(29,19))

# # #

# Comparison with original papers:

# Risk difference:

# Risk difference, CC

# Continuity corrected interval:

# 1.Comparison with results presented in Newcombe (1998),
# Table II, page 877, 10. Score, CC
# column 1 (a): 56/70-48/80: [0.0441; 0.3559]

Prop.diff(x=c(56,70-56),y=c(48,80-48), alternative="two.sided",
  conf.level=0.95, CImethod="CC")

# I. Risk difference, NHS

# Newcombes Hybrid Score interval:

# 1.Comparison with results presented in Newcombe (1998),
# Table II, page 877, 10. Score, noCC
# column 1 (a): 56/70-48/80: [0.0524; 0.3339]

Prop.diff(x=c(56,70-56),y=c(48,80-48), alternative="two.sided",
  conf.level=0.95, CImethod="NHS")

Prop.diff(x=c(56,70-56),y=c(48,80-48), alternative="greater",
  conf.level=0.975, CImethod="NHS")

Prop.diff(x=c(56,70-56),y=c(48,80-48), alternative="less",
  conf.level=0.975, CImethod="NHS")

```

```
# 2.Comparison with results presented in Newcombe (1998),
# Table II, page 877, 10. Score, noCC
# column 2 (b): 9/10-3/10: [0.1705; 0.8090]

Prop.diff(x=c(9,1),y=c(3,7), alternative="two.sided",
  conf.level=0.95, CImethod="NHS")

# 3.Comparison with results presented in Newcombe (1998),
# Table II, page 877, 10. Score, noCC
# column 2 (h): 10/10-0/10: [0.6075; 1.000]

Prop.diff(x=c(10,0),y=c(0,10), alternative="two.sided",
  conf.level=0.95, CImethod="NHS")

# II. Risk ratio, Score
# Score interval according to Gart and Nam (1988)

# 1.Comparison with results presented in Gart and Nam (1998),
# Section 5 (page 327), Example 1
# x1/n1=8/15 x0/n0=4/15:
# Log: [0.768, 4.65]
# Score: [0.815; 5.34]

# Log (GNC)
Prop.ratio(x=c(8,7),y=c(4,11), alternative="two.sided",
  conf.level=0.95, CImethod="GNC")

# Score (Score)
Prop.ratio(x=c(8,7),y=c(4,11), alternative="two.sided",
  conf.level=0.95, CImethod="Score")

Prop.ratio(x=c(8,7),y=c(4,11), alternative="less",
  conf.level=0.975, CImethod="Score")

Prop.ratio(x=c(8,7),y=c(4,11), alternative="greater",
  conf.level=0.975, CImethod="Score")

# 2.Comparison with results presented in Gart and Nam (1998),
# Section 5 (page 328), Example 2
# x1/n1=6/10 x0/n0=6/20:
# Log: [0.883, 4.32]
# Score: [0.844; 4.59]

# Log (GNC)
Prop.ratio(x=c(6,4),y=c(6,14), alternative="two.sided",
  conf.level=0.95, CImethod="GNC")

# Score (Score)
Prop.ratio(x=c(6,4),y=c(6,14), alternative="two.sided",
  conf.level=0.95, CImethod="Score")
```

pairwiseMEP

Wrapper to compute confidence intervals for multiple endpoints

Description

This is a test version! Computes confidence intervals for pair wise comparisons of groups (assuming independent observations) for multiple endpoints. The methods available in pairwiseCI for continuous and count data can be called. Methods for binary data are currently not available. NOTE: Although multiple endpoints and multiple group wise comparisons are considered, there is no adjustment for multiplicity implemented in this function!

Usage

```
pairwiseMEP(x, ...)

## S3 method for class 'data.frame':
pairwiseMEP(x, ep, f,
  control = NULL, conf.level = 0.95,
  alternative = c("two.sided", "less", "greater"),
  method = "Param.diff", ...)
```

Arguments

x	a data.frame
ep	a vector of character strings, naming the variables in x which are the response variables (endpoints) of interest
f	a single character string, naming a factor variable in data which splits the dataset into treatment groups
control	optionally, a single character string, naming a factor level in variable f, which shall be considered as control group; if omitted (default) all pairwise comparisons are computed
conf.level	a single numeric between 0.5 and 1, specifying the local confidence level of the single confidence intervals
alternative	a single character string, one of 'two.sided', 'less', 'greater'
method	a vector of character strings, specifying the method for computation of the confidence intervals, see pairwiseCImethodsCont and pairwiseCImethodsCount for possible options; must have length 1 or the same length as ep!
...	further arguments to be passed to pairwiseCI , options are listed in pairwiseCImethodsCont and pairwiseCImethodsCount

Details

Calls [pairwiseCI](#).

Value

<code>conf.int</code>	a list with one element for each element in <code>ep</code> , containing the estimates, lower and upper limits and the comparison names and by levels in the format of a <code>data.frame</code>
<code>data</code>	as input <code>x</code>
<code>ep</code>	as input
<code>f</code>	as input
<code>control</code>	as input
<code>conf.level</code>	as input
<code>alternative</code>	as input
<code>method</code>	as input

See Also

The result can be plotted: `plotCI.pairwiseMEP`, and coerced to a `data.frame`: `as.data.frame.pairwiseMEP`

Examples

```
x1<-rnorm(120,20,2)
x2<-rnorm(120,100,8)
x3<-rpois(120,10)
x4<-rbinom(120,mu=10, size=10)
A<-rep(c("a1","a2","a3"), c(40,40,40))
B<-rep(rep(c("b1","b2","b3","b4"), c(10,10,10,10)), times=3)
dat<-data.frame(x1=x1,x2=x2,x3=x3,x4=x4,A=A, B=B)

test<-pairwiseMEP(x=dat, ep=c("x1","x2","x3", "x4"), f="A", by="B")
test

plotCI(test)
```

pairwiseTest

Wrapper to calculate unadjusted p-values for pairwise comparisons

Description

Calculation of raw p-values for pairwise comparisons of several groups. The data can be split by additional factors. Any test function can be used, that takes two samples `x,y` as input and returns a list containing the p.value in an element named `p.value`. The output of this function might be further processed using `p.adjust` in order to adjust for multiple comparisons.

Usage

```
pairwiseTest(formula, data, by = NULL,
             alternative = "two.sided", method = "t.test",
             control = NULL, ...)
```

Arguments

<code>formula</code>	a formula specifying the response and the factor variable: <i>response ~ factor</i>
<code>data</code>	a data frame, containing the variables specified in <i>formula</i>
<code>by</code>	optional vector of character strings, defining factors by which to split the data set. Then, pairwise comparisons are performed separately for each level of the specified factors.
<code>alternative</code>	character string, defining whether the direction of the alternative hypothesis, passed to the function defined in <i>method</i>
<code>method</code>	character string, giving the name of the function, which shall be used to calculate local p-values. Any function, taking two vectors <i>x</i> , and <i>y</i> as first arguments and returning a list with the p.value in a list element named <i>p.value</i> can be specified.
<code>control</code>	optional character string, defining the name of a control group. Must be one of the levels of the factor variable defined in <i>formula</i> . By default <code>control=NULL</code> , then all pairwise comparisons between the levels of the factor variable are computed.
<code>...</code>	Arguments to be passed the function defined in <i>method</i>

Details

This function splits the response variable according to the factor(s) specified in *by*, and within each subset according to the grouping variable specified in *formula*. The function specified in *method* is called to calculate a p.value for all pairwise comparisons of between the subsets, within each level of *by*. The p-values are NOT adjusted for multiple hypothesis testing.

For binomial proportions, only *"Prop.test"* can be specified in the argument *method*; For continuous variables, any function can be specified, which takes *x* and *y* as first arguments, and returns a list containing a list containing the appropriate p-value in the element named *p.value* (as do the functions of class *"htest"*). See the examples for details.

Value

A named list with elements

<code>byout</code>	a list, containing the output of <code>pairwiseTestint</code> for each level of <i>by</i> , i.e. a data.frame containing with columns <i>p.value</i> , <i>compnames</i> <i>groupx</i> , <i>groupy</i>
<code>byname</code>	a character vector containing the names of the levels of the factors specified in <i>by</i>
<code>alternative</code>	a character string
<code>method</code>	a character string, name of the function used

```

control      a character string
by           vector of character strings, same as argument by
...         further arguments that were passed to FUN

```

Author(s)

Frank Schaarschmidt

See Also

You can use [summary.pairwiseTest](#) to calculate multiplicity adjusted p-values from the output of `pairwiseTest`.

The following methods provide multiplicity adjusted p-values for various situations: [pairwise.t.test](#), [pairwise.prop.test](#), [p.adjust](#), [summary.glm\(multcomp\)](#), [simtest.ratio\(mratios\)](#)

Examples

```

#####
# The rooting example:
# Calculate confidence intervals for the
# difference of proportions between the 3 doses of IBA,
# separately for 4 combinations of "Age" and "Position".
# Note: we pool over Rep in that way. Whether this makes
# sense or not, is decision of the user.

data(rooting)

# Pairwise Chi-square tests:

aproots<-pairwiseTest(cbind(root, noroot) ~ IBA,
  data=rooting, by=c("Age", "Position"), method="Prop.test")

aproots

# With Holm adjustment for multiple hypotheses testing:

summary(aproots, p.adjust.method="holm")

#####

data(Oats)

apc <- pairwiseTest(yield ~ nitro, data=Oats,
  by="Variety", method="wilcox.test")

apc

summary(apc)
summary(apc, p.adjust.method="holm")

```

```

# # many to one comparisons, with variety Marvellous as control,
# for each level of nitro separately:

m2l <- pairwiseTest(yield ~ Variety, data=Oats,
  by="nitro", method="perm.test", control="Marvellous")
m2l

#####

set.seed(1234)

resp<-rnorm(n=100,
  mean=rep(c(2,5,5,5,5,10,18,5,5,5),each=10),
  sd=rep(c(1,2,2,2,3,5,5,2,2,2), each=10)
)

noise<-rbinom(n=100, size=1, prob=0.05)

resp<-resp + noise*rnorm(n=100,mean=10, sd=10)

fact<-as.factor(rep(LETTERS[1:10], each=10))
data<-data.frame(resp=resp, fact=fact)

boxplot(resp~fact)

# All pairwise comparisons using Welch-tests

aptestW<-pairwiseTest(resp~fact, data=data, method="t.test",
  var.equal=FALSE)
aptestW
summary(aptestW, p.adjust.method="holm", letters=TRUE)

# All pairwise comparisons using Wilcoxon tests

aptestWi<-pairwiseTest(resp~fact, data=data, method="wilcox.test")
aptestWi
summary(aptestWi, p.adjust.method="holm", letters=TRUE)

# All pairwise comparisons using Permutation tests
# from library exactRankTests

aptestP<-pairwiseTest(resp~fact, data=data, method="perm.test")
aptestP
summary(aptestP, p.adjust.method="holm", letters=TRUE)

#####
# Petersens bean example:

data(bean)
boxplot(yield ~ P*T, data=bean)
bean$trt<-paste(bean$P, bean$T, sep="")
bean
# There might be interaction AND heterogeneity of variances.

```

```

# There are definitely smarter ways to analyze these data.
# But a simple-minded way is:

pairwiseTest(yield~trt,data=bean, method="var.test")

# or
# Might there be different variances between the
# comparisons of interest?

pairwiseTest(yield~P,data=bean, by="T", control="P0",
  method="var.test")

# Which Phosphor doses lead to increase of yield
# compared to P0, separate for the two types?

compP<-pairwiseTest(yield~P,data=bean, by="T",
  control="P0", method="t.test", var.equal=FALSE,
  alternative="greater")

# P-values adjusted according to Holm:

summary(compP, p.adjust.method="holm")

# confidence intervals for the same problem,
# using Bonferroni-adjustment:

CIcompP<-pairwiseCI(yield~P, data=bean, by="T",
  control="P0", method="Param.diff", var.equal=FALSE,
  alternative="greater", conf.level=1-0.05/4)

plot(CIcompP)

```

pairwiseTestInt *Internal functions for pairwiseTest*

Description

Only for internal use by pairwiseTest. Two different functions for data representable as a two numeric vectors (pairwiseTestCont) and data representable as matrix with two columns (pairwiseTest-Prop) as created can be done with a formula like *cbind(successes, failures) ~ group*. Functions that split up a data.frame according to one factor, and perform all pairwise comparisons and comparisons to control among the levels of the factor by calling functions that can deal with two vectors x and y or the one documented in ?Prop.test.

Usage

```
pairwiseTestCont(formula, data, alternative="two.sided", control=NULL, method, ...)
```

```
pairwiseTestProp(formula, data, alternative="two.sided", control=NULL, method, ...)
```

Arguments

<code>formula</code>	a formula specifying the response and the factor variable: <i>response</i> ~ <i>factor</i>
<code>data</code>	a data frame, containing the variables specified in <i>formula</i>
<code>alternative</code>	character string, defining whether the direction of the alternative hypothesis, passed to the function defined in <i>method</i>
<code>method</code>	character string, giving the name of the function, which shall be used to calculate local p-values. Any function, taking two vectors <i>x</i> , and <i>y</i> as first arguments and returning a list with the p.value in a list element named <i>p.value</i> can be specified.
<code>control</code>	optional character string, defining the name of a control group. Must be one of the levels of the factor variable defined in <i>formula</i> . By default <code>control=NULL</code> , then all pairwise comparisons between the levels of the factor variable are computed.
<code>...</code>	Arguments to be passed the function defined in <i>method</i>

Details

For internal use in `pairwiseTest`.

Value

a data.frame containing the columns

<code>p.value</code>	numeric vector: the p.values
<code>compnames</code>	character vector: the names of the comparisons performed
<code>groupx</code>	character vector: the names of the first group
<code>groupy</code>	character vector: the names of the second group

See Also

[pairwiseTest](#) for a user level function, and [pairwise.t.test](#), [pairwise.prop.test](#), [p.adjust](#) for further functions to calculate multiplicity adjusted p-values.

plot.pairwiseCI *Plotting the output of pairwiseCI*

Description

Easy method for plotting estimates and confidence bounds calculated using [pairwiseCI](#).

Usage

```
## S3 method for class 'pairwiseCI':
plot(x,
     CIvert=NULL, CIlty = 1, CIlwd=1, CICex=1,
     H0line=NULL, H0lty=1, H0lwd=1,
     main=NULL, ylab="", xlab="",
     ... )
```

Arguments

x	an object of class "pairwiseCI", the output of function pairwiseCI
CIvert	logical, whether confidence intervals shall be plotted vertical if <i>CIvert=TRUE</i> and horizontal if <i>CIvert=FALSE</i>
CIlty	integer, giving the line type of the CI, as documented for <i>cex</i> in <i>?par</i>
CIlwd	integer, giving the line width of the CI, as documented for <i>lwd</i> in <i>?par</i>
CICex	numerical value giving the size of CI symbols relative to the default value, see <i>cex</i> in <i>?par</i>
H0line	Value to be plotted as vertical or horizontal line, depending on the value of <i>CIvert</i>
H0lty	integer, giving the line type of the CI, as documented for <i>lty</i> in <i>?par</i>
H0lwd	integer, giving the line width of the CI, as documented for <i>lwd</i> in <i>?par</i>
main	as <i>main</i> in <i>plot</i>
ylab	label of y-axis as <i>ylab</i> in <i>plot</i> , default is no label
xlab	label of x-axis as <i>ylab</i> in <i>plot</i> , default is no label
...	Further arguments to be passed to <i>plot</i> . Note, that for adjusting character size in the axes, one must currently adjust <i>par(cex.axis)</i> , and some arguments as <i>las</i> , <i>at</i> , <i>labels</i> are defined internally.

Author(s)

Frank Schaarschmidt

Examples

```

data(Oats)

output <- pairwiseCI(yield ~ Block, data=Oats,
  by="nitro",method="Param.diff", control="I")

# default plot for difference methods:
plot(output)

# some small changes:
plot(output, CIvert=TRUE, H0line=c(-2,0,2), H0lty=c(2,1,2))

output <- pairwiseCI(yield ~ Block, data=Oats,
  by="nitro", method="Param.ratio", control="I")

# default plot for ratio methods:
plot(output)

# some small changes:
plot(output, CIvert=FALSE, H0line=c(0.7, 1, 1/0.7),
  H0lty=c(3,2,3))

```

plotCI.pairwiseMEP *Plot confidence intervals calculated by "pairwiseMEP"*

Description

Creates plot of confidence intervals calculated by calling "pairwiseMEP".

Usage

```

## S3 method for class 'pairwiseMEP':
plotCI(x, whichep = NULL, ...)

```

Arguments

x	an object of class ' <i>pairwiseMEP</i> ' as can be created by calling <code>pairwiseMEP</code>
whichep	an optional vector of character strings (or integers); specifying the names (or indices in the element <i>conf.int</i> of the list returned by <i>pairwiseMEP</i>) of those response variables for which the confidence intervals shall be plotted
...	further arguments to be passed to <i>plotCI</i> in package MCPAN , see <i>?plotCI</i> for details

Value

A plot.

Examples

```
x1<-rnorm(120,20,2)
x2<-rnorm(120,100,8)
x3<-rpois(120,10)
x4<-rnbino(120,mu=10, size=10)
A<-rep(c("a1","a2","a3"), c(40,40,40))
B<-rep(rep(c("b1","b2","b3","b4"), c(10,10,10,10)), times=3)
dat<-data.frame(x1=x1,x2=x2,x3=x3,x4=x4,A=A, B=B)

test<-pairwiseMEP(x=dat, ep=c("x1","x2","x3","x4"), f="A", by="B")

plotCI(test, whichep=c("x1","x2"), lines=c(-5,5))

plotCI(test, whichep=c(3,4))
```

```
print.pairwiseCI Print function for "pairwiseCI"
```

Description

Print out confidence intervals calculated using pairwiseCI.

Usage

```
## S3 method for class 'pairwiseCI':
print(x, digits=4, ...)
```

Arguments

x	an object of class "pairwiseCI", the output of function pairwiseCI()
digits	integer, to which decimal output shall be rounded
...	further arguments to be passed to <i>print</i>

```
print.pairwiseTest Print function for "pairwiseTest"
```

Description

Print function for pairwiseTest objects

Usage

```
## S3 method for class 'pairwiseTest':
print(x, digits = 4, ...)
```

Arguments

<code>x</code>	an object of class "pairwiseTest"
<code>digits</code>	digits for rounding
<code>...</code>	further arguments to be passed to <i>print</i>

```
print.summary.pairwiseCI
      Print function for "summary.pairwiseCI"
```

Description

Print function for `summary.pairwiseCI`

Usage

```
## S3 method for class 'summary.pairwiseCI':
print(x, ...)
```

Arguments

<code>x</code>	an object of class "summary.pairwiseCI", created by the function <code>summary.pairwiseCI</code>
<code>...</code>	further arguments to be passed to <i>print</i>

```
print.summary.pairwiseTest
      Print function for "summary.pairwiseTest"
```

Description

Print function for `summary.pairwiseCI`

Usage

```
## S3 method for class 'summary.pairwiseTest':
print(x, ...)
```

Arguments

<code>x</code>	an object of class "summary.pairwiseTest", created by the function <code>summary.pairwiseTest</code>
<code>...</code>	further arguments to be passed to <i>print</i>

 profileDG

Construct a (quasi-) likelihood-profile

Description

Construct a (quasi-) likelihood-profile based on a glm-fit. For internal use for functions constructing profile-likelihood confidence intervals.

Usage

```
profileDG(fit, steps = 100, wh = 1:p)
```

Arguments

fit	an object of class "glm" or "negbin"
steps	a single integer value, the number of steps for the profile
wh	wh

Details

An adaptation of the code in *profile.glm*. Will work also for the case that only 0-values occur in one group. For internal use

Value

An object of class "*profile.glm*", "*profile*".

Author(s)

Daniel Gerhard

 Prop.test

Wrapper to prop.test(stats)

Description

Only for internal use in pairwiseTest.

Usage

```
Prop.test(x, y, alternative = "two.sided", ...)
```

Arguments

<code>x</code>	a vector of success and failure in sample <code>x</code> , or a data.frame with a column of successes and a column of failures, then <code>colSums</code> are used.
<code>y</code>	a vector of success and failure in sample <code>y</code> , or a data.frame with a column of successes and a column of failures, then <code>colSums</code> are used.
<code>alternative</code>	character string defining the alternative hypothesis
<code>...</code>	arguments to be passed to <code>prop.test(stats)</code>

Details

Just a wrapper function to call `prop.test(stats)`. If `x`, `y` are data.frames containing two columns taken to be counts of successes and counts of failures, columnwise sums of `x,y` are calculated. The total number of successes and the total number of trials is then passed to `prop.test`.

Value

An object of class "htest", as defined by `prop.test(stats)`

See Also

`prop.test`, and `pairwise.prop.test` in **stats**

Examples

```
# If input is a data.frame:

set.seed(1234)

trials=rep(20,8)
success <- rbinom(n=8, size=trials,
  prob=c(0.2,0.2,0.2,0.2, 0.3,0.3,0.3,0.3))
failure <- trials-success

f<-as.factor(rep(c("group1", "group2"), each=4))

data<-data.frame(success=success, failure=failure, f=f)

g1<-subset(data, f=="group1")[,c("success","failure")]
g2<-subset(data, f=="group2")[,c("success","failure")]

g1
g2

# Prop.test calculates the columnwise sums and calls prop.test stats:

Prop.test(x=g1, y=g2)

# should be the same as:
```

```
CS1<-colSums (g1)
CS2<-colSums (g2)

CS1
CS2

prop.test(x=c(CS1[1], CS2[1]), n=c(sum(CS1), sum(CS2)))
```

repellent

Repellent effect of sulphur in eight concentrations

Description

Sugar solutions were presented to certain number of bees. To assess the repellent effect of the fungicide sulphur of bees, increasing concentration of sulphur was added to the sugar solutions. The decrease of sugar solutions (i.e. uptake by bees) was measured. Low values of decrease therefore can be interpreted as high repellent effect of the sulphur concentration. Assumed to be a completely randomized design.

Usage

```
data(repellent)
```

Format

A data frame with 64 observations on the following 2 variables.

decrease a numeric vector, the absolut decrease of sugar solutions from begin to end of the experiment

treatment a factor with levels A B C D E F G H, the concentration of sulphur

Examples

```
data(repellent)
boxplot(decrease ~ treatment, data=repellent)
```

 rooting

Rooting (success/failure) of plants in a 3-factorial field trial

Description

Part of an experiment on propagation of plant genera Acer and Pyrus. Cuttings were taken from motherplants of age 5 and age 20, from top or base, and were treated with 0, 0.5 and 2 percent IBA to induce rooting. Treatments were arranged in a completely randomized design, among other variables, the number of cuttings with and without roots was recorded.

Usage

```
data(rooting)
```

Format

A data frame with 48 observations on the following 6 variables.

Age a numeric vector: age of mother plants

Position a factor with levels B and T, for "base" and "top" cuttings

IBA a numeric vector, specifying the concentration of IBA

Rep a numeric vector, number of replication

root a numeric vector, number of cuttings with successful rooting, out of 12 trials

noroot a numeric vector, number of cuttings showing no rooting, out of 12 trials

Source

Data taken from Msc thesis by Dawit Mamushet Yifru, Institute of Floriculture, Tree Nursery Science and Plant Breeding, University of Hannover, 2005.

Examples

```
data(rooting)

rooting$IBAf<-as.factor(rooting$IBA)
rooting$Rep<-as.factor(rooting$Rep)

fitB<-glm(cbind(root,noroot)~Rep+(Age + Position + IBA)^2,
  data=rooting, family=binomial)

fitQB<-glm(cbind(root,noroot)~Rep+(Age + Position + IBA)^2,
  data=rooting, family=quasibinomial)

summary(fitB)
summary(fitQB)

anova(fitB, test="Chisq")
anova(fitQB, test="F")
```

sodium	<i>Sodium contents in transgenic and isogenic corn</i>
--------	--

Description

Sodium was measured in transgenic corn and the original isogenic corn variety.

Usage

```
data(sodium)
```

Format

A data frame with 12 observations on the following 2 variables.

Treatment a factor with levels `transgenic` `isogenic`

Sodiumcontent a numeric vector

Source

Oberdoerfer, R.B. Example dataset from composition analyses of genetically modified oilseed rape seeds. 2003; BCS GmbH.

Examples

```
data(sodium)
boxplot(Sodiumcontent ~Treatment, data=sodium)
```

<code>summary.pairwiseCI</code>	<i>Summary function for pairwiseCI</i>
---------------------------------	--

Description

Creates a list of data.frames from the output of `pairwiseCI`

Usage

```
## S3 method for class 'pairwiseCI':
summary(object, digits = 4, ...)
```

Arguments

<code>object</code>	An object of class <i>"pairwiseCI"</i> , created using the function <code>pairwiseCI</code>
<code>digits</code>	number of digits for rounding of results
<code>...</code>	Currently not used.

Value

A list.

See Also

link{as.data.frame.pairwiseCI}

Examples

```
data(rooting)

rootRR<-pairwiseCI(cbind(root,noroot) ~ IBA,
  data=rooting, by="Age", method="Prop.ratio")

# after calling summary,
# extracting parts of the output is easier:

srootRR<-summary(rootRR)

srootRR$'20'$conf.int$upper
```

```
summary.pairwiseTest
```

Summary function for "pairwiseTest"

Description

Creates a data.frame from the output of pairwiseTest, allows to adjust raw p-values by methods implemented in p.adjust.

Usage

```
## S3 method for class 'pairwiseTest':
summary(object, digits = 4,
  p.adjust.method = "none", ...)
```

Arguments

object	An object of class "pairwiseTest", created using the function <code>pairwiseTest</code>
digits	number of digits for rounding of results
p.adjust.method	Method to adjust p-values for multiple hypothesis testing, see options in <code>p.adjust.method</code> in stats . The default in this function is "none", resulting in unadjusted p-values
...	Currently not used.

Details

Coerces the raw p-values and the corresponding group levels to a data.frame and applies p.adjust to it.

Value

A dataframe, with columns

p.val.raw	raw p-values
p.val.adj	adjusted p-values, according to the method specified in <i>p.adjust.method</i>
comparison	the calculated differences or ratios of parameters
groupx	levels of group x
groupy	levels of group y

and possibly further columns containing levels of by.

Examples

```
data(Oats)
apOats<-pairwiseTest(yield~nitro, data=Oats,
  by="Variety", method="t.test", var.equal=FALSE)
apOats

# summary just creates a data.frame from the output
summary(apOats)

# an allows application of p.adjust
# on the p.values:

summary(apOats, p.adjust.method="holm")

# See ?p.adjust.methods for the methods available.
```

Index

*Topic **datasets**

- bean, 6
- behenic, 7
- cabbage, 8
- dieldrin, 8
- Oats, 9
- repellent, 41
- rooting, 42
- sodium, 43

*Topic **hplot**

- plot.pairwiseCI, 34
- plotCI.pairwiseMEP, 36

*Topic **htest**

- pairwiseCI, 10
- pairwiseCI-package, 2
- pairwiseCIInt, 16
- pairwiseCImethodsCont, 18
- pairwiseCImethodsCount, 21
- pairwiseCImethodsProp, 23
- pairwiseMEP, 27
- pairwiseTest, 29
- pairwiseTestInt, 33
- profileDG, 39
- Prop.test, 39
- summary.pairwiseTest, 44

*Topic **misc**

- as.data.frame.pairwiseCI, 4
- as.data.frame.pairwiseMEP, 5

*Topic **package**

- pairwiseCI-package, 2

*Topic **print**

- print.pairwiseCI, 37
- print.pairwiseTest, 37
- print.summary.pairwiseCI, 38
- print.summary.pairwiseTest, 38
- summary.pairwiseCI, 43
- summary.pairwiseTest, 44

as.data.frame.pairwiseCI, 3, 13

as.data.frame.pairwiseMEP, 5, 29

- bean, 6
- behenic, 7

- cabbage, 8

- dieldrin, 8

HD.diff(*pairwiseCImethodsCont*), 18

HD.ratio(*pairwiseCImethodsCont*), 18

HL.diff(*pairwiseCImethodsCont*), 18

HL.ratio(*pairwiseCImethodsCont*), 18

Lognorm.diff(*pairwiseCImethodsCont*), 18

Lognorm.ratio(*pairwiseCImethodsCont*), 18

Median.diff(*pairwiseCImethodsCont*), 18

Median.ratio(*pairwiseCImethodsCont*), 18

Negbin.ratio(*pairwiseCImethodsCount*), 21

- Oats, 9

p.adjust, 30, 34

pairwise.prop.test, 30, 34

pairwise.t.test, 30, 34

pairwiseCI, 4, 10, 17, 28, 34, 35, 43

pairwiseCI-package, 2

pairwiseCICont(*pairwiseCIInt*), 16

pairwiseCIInt, 16

pairwiseCImethodsCont, [11](#), [16](#), [17](#), [18](#),
[28](#)
pairwiseCImethodsCount, [21](#), [28](#)
pairwiseCImethodsProp, [11](#), [16](#), [17](#), [23](#)
pairwiseCIProp (pairwiseCIInt), [16](#)
pairwiseMEP, [5](#), [6](#), [27](#), [36](#)
pairwiseTest, [29](#), [34](#), [44](#)
pairwiseTestCont
 (pairwiseTestInt), [33](#)
pairwiseTestInt, [33](#)
pairwiseTestProp
 (pairwiseTestInt), [33](#)
Param.diff
 (pairwiseCImethodsCont), [18](#)
Param.ratio
 (pairwiseCImethodsCont), [18](#)
plot.pairwiseCI, [13](#), [34](#)
plotCI.pairwiseMEP, [29](#), [36](#)
Poisson.ratio
 (pairwiseCImethodsCount),
 [21](#)
print.pairwiseCI, [12](#), [37](#)
print.pairwiseTest, [37](#)
print.summary.pairwiseCI, [38](#)
print.summary.pairwiseTest, [38](#)
profileDG, [39](#)
Prop.diff
 (pairwiseCImethodsProp), [23](#)
Prop.or (pairwiseCImethodsProp),
 [23](#)
Prop.ratio
 (pairwiseCImethodsProp), [23](#)
Prop.test, [39](#)

Quasipoisson.ratio
 (pairwiseCImethodsCount),
 [21](#)

repellent, [41](#)
rooting, [42](#)

sodium, [43](#)
summary.pairwiseCI, [13](#), [17](#), [38](#), [43](#)
summary.pairwiseTest, [4](#), [30](#), [38](#), [44](#)