Package 'qountstat'

October 16, 2026

Type Package

Title Statistical Analysis of Count Data and Quantal Data

Version 0.2.0

Maintainer Benjamin Daniels <benjamin.daniels@rwth-aachen.de>

Description Methods for statistical analysis of count data and quantal data.
For the analysis of count data an implementation of the Closure Principle Computational Approach Test (``CPCAT'') is provided (Lehmann, R et al. (2016) <doi:10.1007/s00477-015-1079-4>), as well as an implementation of a ``Dunnett GLM'' approach using either a Quasi-Poisson, a negative binomial or a Poisson regression (Hothorn, L, Kluxen, F (2020) <doi:10.1101/2020.01.15.907881>).
For the analysis of quantal data an implementation of the Closure Principle Fisher–Freeman–Halton test (``CPFISH'') is provided (Lehmann, R et al. (2018) <doi:10.1007/s00477-017-1392-1>). P-values and no/lowest observed (adverse) effect concentration values are calculated.
All implemented methods include further functions to evaluate the power and the minimum detectable difference using a bootstrapping approach.
A model-agnostic, simulation-calibrated diagnostic for overdispersion in grouped count data is provided to support the choice between the implemented methods.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports MASS, multcomp

Suggests knitr, rmarkdown, testthat

RoxygenNote 7.3.2

Depends R (>= 2.10)

NeedsCompilation no

Author Benjamin Daniels [cre, aut] (ORCID:
<https://orcid.org/0000-0002-8344-6141>),
Christian Dietrich [aut],
Thomas Graeff [aut],
Magnus Wang [aut]

Repository CRAN

Date/Publication 2026-10-16 16:20:02 UTC

Contents

calibrate.c.alpha . . . 2
CP.hypotheses . . . 3
CPCAT . . . 4
CPCAT.bMDD . . . 6
CPCAT.Poisson.sub.test . . . 7
CPCAT.Poisson.test . . . 8
CPCAT.power . . . 8
CPFISH . . . 10
CPFISH.bMDD . . . 11
CPFISH.contingency.table . . . 12
CPFISH.power . . . 13
Daphnia.counts . . . 14
Dunnett.GLM . . . 15
Dunnett.GLM.bMDD . . . 18
Dunnett.GLM.power . . . 20
MSSR.statistic . . . 22
Multi.group.test.bMDD . . . 22
Multi.group.test.power . . . 24
Overdispersion.test . . . 25
run.overdispersion.diagnostic . . . 27

Index **29**

`calibrate.c.alpha` *Monte Carlo calibration of the decision threshold*

Description

Simulates the null distribution of the MSSR statistic for a given design. Internal helper of Overdispersion.test.

Usage

```
calibrate.c.alpha(
  mu.vector,
  n.vector,
  bootstrap.runs,
  pretest.mode = "tolerant",
  tolerance.factor = 2,
  zero.group.mean = 0.5
)
```

Arguments

`mu.vector`	Vector with the expected value per treatment group
`n.vector`	Vector with the number of observations per treatment group
`bootstrap.runs`	Number of Monte Carlo simulations
`pretest.mode`	Reference distribution ("tolerant" or "strict")
`tolerance.factor`	Tolerated variance-to-mean ratio (only used if pretest.mode = "tolerant")
`zero.group.mean`	Value that replaces a group mean of zero in the denominator

Value

Numeric vector with the simulated null distribution of the test statistic

---

`CP.hypotheses` *CP hypotheses*

Description

Function to generate hypotheses for the Closure Principle concept using 0/1 contrast matrices

Usage

```
CP.hypotheses(n, treatment.names = NULL)
```

Arguments

`n`	Number of groups
`treatment.names`	Optional vector with names of treatment groups

Value

Contrast matrix (all 0/1 combinations)

CPCAT *CPCAT*

Description

When conducting statistical tests with multiple treatments, such as a control group and increasing concentrations of a test substance, ANOVA and parametric post-hoc tests (e.g. Dunnett's test) are commonly used. However, these tests require the assumptions of homogeneous variances and normally distributed data. For count data (e.g. counts of animals), these assumptions are typically violated, as the data are usually Poisson-distributed. Additionally, multiple testing using post-hoc tests can lead to alpha-inflation. To address these issues, CPCAT was proposed by Lehmann et al. (2016). CPCAT has two components. The first is the Closure Principle (CP) developed by Bretz et al. (2010), which aims to eliminate alpha-inflation. CP applies a stepwise approach to identify at which concentration effects begin to occur. The second part of CPCAT is the actual significance test, CAT (Computational Approach Test; introduced by Chang et al., 2010), which uses a test based on the Poisson distribution rather than a parametric test based on normal distribution assumptions. For details on the structure of the input data, please refer to the dataset 'Daphnia.counts' provided alongside this package.

Usage

```
CPCAT(
  groups,
  counts,
  control.name = NULL,
  bootstrap.runs = 10000,
  hampel.threshold = NULL,
  use.fixed.random.seed = NULL,
  get.contrasts.and.p.values = FALSE,
  show.output = TRUE,
  overdispersion.test = NULL
)
```

Arguments

`groups`	Group vector
`counts`	Vector with count data
`control.name`	Character string with control group name (optional)
`bootstrap.runs`	Number of bootstrap runs
`hampel.threshold`	Threshold for Hampel identifier (measure for over-/underdispersion). If NULL (default) the Hampel identifier is not evaluated. The Hampel identifier compares the difference between variance and mean of each treatment group with a fixed threshold. Since the sampling variability of this difference grows proportionally to the mean while the threshold does not, the identifier becomes increasingly

sensitive at higher count levels and flags Poisson-distributed data as overdispersed far too often (Daniels et al., 2026). It is therefore switched off by default and should only be enabled for backward compatibility or at low count levels. See `Overdispersion.test` for a scale-invariant alternative, and note that no dispersion diagnostic should be relied on alone: the data should always be inspected graphically as well, for example by plotting the group variances against the group means together with the identity line.

`use.fixed.random.seed`
: Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.

`get.contrasts.and.p.values`
: Get each row of the contrast matrices evaluated

`show.output`
: Show/hide output

`overdispersion.test`
: Add the result of the overdispersion pre-test to the information object. Use NULL or FALSE (default) to switch the pre-test off, TRUE to run it with the default settings of `Overdispersion.test`, or a named list with any of the arguments alpha, bootstrap.runs, pretest.mode, tolerance.factor, use.fixed.random.seed, include.zero.groups and zero.group.mean to control it. Enabling the pre-test does not change the CPCAT results, since the state of the random number generator is restored afterwards.

Value

R object with results and information from CPCAT calculations

References

Daniels, B.; Kotschik, P.; Walter-Rohde, S.; Graeff, T. (2026): When Counts Deviate: Robust Overdispersion Diagnostics for Ecotoxicity Data. Poster presented at the 36th SETAC Europe Annual Meeting, 17-21 May 2026, Maastricht, The Netherlands.

Bretz, F.; Hothorn, T.; Westfall, P. (2010): Multiple comparisons using R. 1st Edition, Chapman and Hall/CRC, New York

Chang, C.-H.; Pal, N.; Lin, J.-J. (2010): A Note on Comparing Several Poisson Means. In: Commun. Stat. Simul. Comput., 2010, 39(8), p. 1605-1627, https://doi.org/10.1080/03610918.2010.508860

Lehmann, R.; Bachmann, J.; Maletzki, D.; Polleichtner, C.; Ratte, H.; Ratte, M. (2016): A new approach to overcome shortcomings with multiple testing of reproduction data in ecotoxicology. In: Stochastic Environmental Research and Risk Assessment, 2016, 30(3), p. 871-882, https://doi.org/10.1007/s00477-015-1079-4

Examples

```
Daphnia.counts # example data provided alongside the package

# Test CPCAT
CPCAT(groups = Daphnia.counts$Concentration,
  counts = Daphnia.counts$Number_Young,
  control.name = NULL,
  bootstrap.runs = 10000,
```

```
  use.fixed.random.seed = 123,  #fixed seed for reproducible results
  get.contrasts.and.p.values = FALSE,
  show.output = TRUE)

# Test CPCAT and add the overdispersion pre-test to the information object
CPCAT(groups = Daphnia.counts$Concentration,
  counts = Daphnia.counts$Number_Young,
  control.name = NULL,
  bootstrap.runs = 1000, # Caution: low number of bootstrap runs for testing
  use.fixed.random.seed = 123,  #fixed seed for reproducible results
  get.contrasts.and.p.values = FALSE,
  show.output = TRUE,
  overdispersion.test = list(bootstrap.runs = 500, use.fixed.random.seed = 123))
```

`CPCAT.bMDD` *CPCAT bootstrap MDD (bMDD)*

Description

The basic idea of the calculation of bootstrap MDD (bMDD) using the CPCAT approach is to shift the lambda parameter of Poisson distribution until there is a certain proportion of results significantly different from the control.

Usage

```
CPCAT.bMDD(
  groups,
  counts,
  control.name = NULL,
  alpha = 0.05,
  shift.step = -0.25,
  bootstrap.runs = 200,
  power = 0.8,
  max.iterations = 1000,
  use.fixed.random.seed = NULL,
  CPCAT.bootstrap.runs = 200,
  show.progress = TRUE,
  show.results = TRUE
)
```

Arguments

`groups`	Group vector
`counts`	Vector with count data
`control.name`	Character string with control group name (optional)
`alpha`	Significance level
`shift.step`	Step of shift (negative as a reduction is assumed)

bootstrap.runs	Number of bootstrap runs
power	Proportion of bootstrap.runs that return significant differences
max.iterations	Max. number of iterations to not get stuck in the while loop
use.fixed.random.seed	Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.
CPCAT.bootstrap.runs	Bootstrap runs within CPCAT method
show.progress	Show progress for each shift of lambda
show.results	Show results

Value

Data frame with results from bMDD analysis

Examples

```
Daphnia.counts # example data provided alongside the package

# Test CPCAT bootstrap MDD

CPCAT.bMDD(groups = Daphnia.counts$Concentration,
    counts = Daphnia.counts$Number_Young,
    control.name = NULL,
    alpha = 0.05,
    shift.step = -1,  # Caution: big step size for testing
    bootstrap.runs = 5, # Caution: low number of bootstrap runs for testing
    power = 0.8,
    max.iterations = 1000,
    use.fixed.random.seed = 123,  #fixed seed for reproducible results
    CPCAT.bootstrap.runs = 10,
    show.progress = TRUE,
    show.results = TRUE)
```

CPCAT.Poisson.sub.test

CPCAT Poisson sub-test

Description

Helper function for CPCAT

Usage

```
CPCAT.Poisson.sub.test(dat, contrast, bootstrap.runs = 10000)
```

Arguments

dat	Data frame to be evaluated
contrast	Contrast matrix
bootstrap.runs	Number of bootstrap runs

Value

p-value for tested data and contrast

CPCAT.Poisson.test *CPCAT Poisson test*

Description

Helper function for CPCAT

Usage

```
CPCAT.Poisson.test(dat, contrastmatrix, bootstrap.runs = 10000)
```

Arguments

dat	Data frame to be evaluated
contrastmatrix	Contrast matrix
bootstrap.runs	Number of bootstrap runs

Value

List of p-values for tested contrast matrices

CPCAT.power *CPCAT power*

Description

The basic idea of CPCAT power calculations is to do parametric bootstrapping for each dose/concentration group and to evaluate the proportion of results significantly different from the control.

Usage

```
CPCAT.power(
  groups,
  counts,
  control.name = NULL,
  alpha = 0.05,
  bootstrap.runs = 200,
  use.fixed.random.seed = NULL,
  CPCAT.bootstrap.runs = 200,
  show.progress = TRUE,
  show.results = TRUE
)
```

Arguments

groups	Group vector
counts	Vector with count data
control.name	Character string with control group name (optional)
alpha	Significance level
bootstrap.runs	Number of bootstrap runs
use.fixed.random.seed	Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.
CPCAT.bootstrap.runs	Bootstrap runs within CPCAT method
show.progress	Show progress for each shift of lambda
show.results	Show results

Value

Data frame with results from power analysis

Examples

```
Daphnia.counts # example data provided alongside the package

# Test CPCAT power
CPCAT.power(groups = Daphnia.counts$Concentration,
     counts = Daphnia.counts$Number_Young,
     control.name = NULL,
     alpha = 0.05,
     bootstrap.runs = 10,  # Caution: low number of bootstrap runs for testing
     use.fixed.random.seed = 123,  #fixed seed for reproducible results
     CPCAT.bootstrap.runs = 10,# Caution: low number of bootstrap runs for testing
     show.progress = TRUE,
     show.results = TRUE)
```

CPFISH *CPFISH*

Description

For quantal data (e.g. survival data, '14 out of 20 animals died') e.g. in the form of a contingency table, CPFISH was proposed by Lehmann et al. (2018). Like CPCAT, CPFISH is based on the Closure Principle (CP), but instead of a bootstrapping approach, a Fisher test is performed for all sub-hypotheses to be analyzed. For details on the structure of the input table, please refer to the dataset 'CPFISH.contingency.table' provided alongside this package.

Usage

```
CPFISH(
  contingency.table,
  control.name = NULL,
  simulate.p.value = TRUE,
  use.fixed.random.seed = NULL,
  show.output = TRUE
)
```

Arguments

`contingency.table`	Matrix with observed data (e.g. survival counts, survival must be in first row)
`control.name`	Character string with control group name (optional)
`simulate.p.value`	Use simulated p-values in Fisher test or not
`use.fixed.random.seed`	Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.
`show.output`	Show/hide output

Value

R object with results and information from CPFISH calculations

References

Lehmann, R.; Bachmann, J.; Karaoglan, B.; Lacker, J.; Polleichtner, C.; Ratte, H.; Ratte, M. (2018): An alternative approach to overcome shortcomings with multiple testing of binary data in ecotoxicology. In: Stochastic Environmental Research and Risk Assessment, 2018, 32(1), p. 213-222, https://doi.org/10.1007/s00477-017-1392-1

Examples

```
CPFISH.contingency.table # example data provided alongside the package

# Test CPFISH
CPFISH(contingency.table = CPFISH.contingency.table,
  control.name = NULL,
  simulate.p.value = TRUE,
  use.fixed.random.seed = 123,  #fixed seed for reproducible results
  show.output = TRUE)
```

CPFISH.bMDD *CPFISH bootstrap MDD (bMDD)*

Description

The basic idea of the calculation of bootstrap MDD (bMDD) using the CPCAT approach is to shift the probability of binomial distribution until there is a certain proportion of results significantly different from the control.

Usage

```
CPFISH.bMDD(
  contingency.table,
  control.name = NULL,
  alpha = 0.05,
  shift.step = -0.01,
  bootstrap.runs = 200,
  power = 0.8,
  max.iterations = 1000,
  simulate.p.value = TRUE,
  use.fixed.random.seed = NULL,
  show.progress = TRUE,
  show.results = TRUE
)
```

Arguments

`contingency.table`	Matrix with observed data (e.g. survival counts, survival must be in first row)
`control.name`	Character string with control group name (optional)
`alpha`	Significance level
`shift.step`	Step of shift (negative as a reduction is assumed)
`bootstrap.runs`	Number of bootstrap runs (draw Poisson data n times)
`power`	Proportion of bootstrap.runs that return significant differences
`max.iterations`	Max. number of iterations to not get stuck in the while loop

`simulate.p.value`
: Use simulated p-values in Fisher test or not

`use.fixed.random.seed`
: Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.

`show.progress`
: Show progress for each shift of the probability

`show.results`
: Show results

Value

Data frame with results from bMDD analysis

Examples

```
CPFISH.contingency.table # example data provided alongside the package

# Test CPFISH bootstrap MDD
CPFISH.bMDD(contingency.table = CPFISH.contingency.table,
     control.name = NULL,
     alpha = 0.05,
     shift.step = -0.1,  # Caution: big step size for testing
     bootstrap.runs = 10,  # Caution: low number of bootstrap runs for testing
     power = 0.8,
     max.iterations = 1000,
     simulate.p.value = TRUE,
     use.fixed.random.seed = 123,  #fixed seed for reproducible results
     show.progress = TRUE,
     show.results = TRUE)
```

CPFISH.contingency.table

CPFISH.contingency.table is a dataset to showcase analyses with CPFISH

Description

CPFISH.contingency.table is a dataset to showcase analyses with CPFISH

Usage

```
CPFISH.contingency.table
```

Format

An object of class `matrix` (inherits from `array`) with 2 rows and 3 columns.

Source

The CPFISH data was artificially created.

Examples

```
data(CPFISH.contingency.table)
```

`CPFISH.power` *CPFISH power*

Description

The basic idea of CPFISH power calculations is to do parametric bootstrapping for each dose/concentration group and to evaluate the proportion of results significantly different from the control.

Usage

```
CPFISH.power(
  contingency.table,
  control.name = NULL,
  alpha = 0.05,
  bootstrap.runs = 200,
  simulate.p.value = TRUE,
  use.fixed.random.seed = NULL,
  show.progress = TRUE,
  show.results = TRUE
)
```

Arguments

`contingency.table`	Matrix with observed data (e.g. survival counts, survival must be in first row)
`control.name`	Character string with control group name (optional)
`alpha`	Significance level
`bootstrap.runs`	Number of bootstrap runs (draw Poisson data n times)
`simulate.p.value`	Use simulated p-values in Fisher test or not
`use.fixed.random.seed`	Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.
`show.progress`	Show progress for each shift of the probability
`show.results`	Show results

Value

Data frame with results from power analysis

Examples

```
CPFISH.contingency.table # example data provided alongside the package

# Test CPFISH power
CPFISH.power(contingency.table = CPFISH.contingency.table,
   control.name = NULL,
   alpha = 0.05,
   bootstrap.runs = 10, # Caution: low number of bootstrap runs for testing
   simulate.p.value = TRUE,
   use.fixed.random.seed = 123,  #fixed seed for reproducible results
   show.progress = TRUE,
   show.results = TRUE)
```

Daphnia.counts *Daphnia.counts is a dataset to showcase analyses with CPCAT and Dunnett.GLM*

Description

Daphnia.counts is a dataset to showcase analyses with CPCAT and Dunnett.GLM

Usage

```
Daphnia.counts
```

Format

An object of class `data.frame` with 60 rows and 3 columns.

Source

The daphnia data was taken from Hothorn and Kluxen (2020).

Examples

```
data(Daphnia.counts)
```

`Dunnett.GLM` *Dunnett.GLM*

Description

When conducting statistical tests with multiple treatments, such as a control group and increasing concentrations of a test substance, ANOVA and parametric post-hoc tests (e.g. Dunnett's test) are commonly used. However, these tests require the assumptions of homogeneous variances and normally distributed data. For count data (e.g. counts of animals), these assumptions are typically violated, as the data are usually Poisson-distributed. The Dunnett.GLM function is based on a GLM followed by a Dunnett test to the model estimates. It was implemented to serve as an alternative approach to CPCAT while using a Quasi-Poisson regression (default), a negative binomial regression or a classical Poisson regression. The basic approach from Hothorn and Kluxen (2020) was adjusted to overcome methodological issues (see description of 'zero.treatment.action parameter'). For details on the structure of the input data, please refer to the dataset 'Daphnia.counts' provided alongside this package.

Usage

```
Dunnett.GLM(
  groups,
  counts,
  control.name = NULL,
  zero.treatment.action = "identity.link",
  show.output = TRUE,
  family = "quasipoisson",
  overdispersion.test = NULL
)
```

Arguments

`groups`	Group vector
`counts`	Vector with count data
`control.name`	Character string with control group name (optional)
`zero.treatment.action`	Method for dealing with treatments only containing zeros (use either "identity.link" or "log(x+1)"). The method is only used if the data set contains dose/concentration groups that exclusively contain zero values (since the basic method provides for a logarithmic transformation of the data averages, it would lead to incorrect results). To deal with this methodological shortcoming, two options were implemented. The 'identity.link' option: the 'identity' link is used in the GLM instead of the 'log' link, i.e. the data are no longer transformed. The 'log(x+1)' option: The 'log' link is retained and 1 is added to each count value at the start of the procedure so that the subsequent log-transformation can be carried out without any problems. Note that both options may slightly distort the results.
`show.output`	Show/hide output

`family` Distribution family used for the GLM (use either "quasipoisson", "nb" or "poisson"). The default "quasipoisson" fits a Quasi-Poisson regression, i.e. the variance is modelled as a constant multiple (the dispersion parameter) of the mean. The option "nb" fits a negative binomial regression via `glm.nb`, i.e. the variance is modelled as mu + mu^2/theta, with the dispersion parameter theta being estimated from the data by maximum likelihood. Both of these options are suited for overdispersed count data. The negative binomial model is a fully parametric likelihood model and is often preferable if the overdispersion increases more than proportionally with the mean, whereas the Quasi-Poisson model makes weaker distributional assumptions. The option "poisson" fits a classical Poisson regression, i.e. the variance is assumed to equal the mean and the dispersion parameter is fixed at 1 rather than estimated. This option is the regression counterpart of CPCAT and is intended for data that are compatible with a Poisson distribution; it should only be used if that assumption has been checked, for example with `Overdispersion.test` and by inspecting the data graphically (see the note below), because standard errors are underestimated and the inference becomes anticonservative if the data are overdispersed. Note that "poisson" and "nb" require (approximately) integer counts, while "quasipoisson" also accepts non-integer values.

`overdispersion.test` Add the result of the overdispersion pre-test to the information object. Use NULL or FALSE (default) to switch the pre-test off, TRUE to run it with the default settings of `Overdispersion.test`, or a named list with any of the arguments alpha, bootstrap.runs, pretest.mode, tolerance.factor, use.fixed.random.seed, include.zero.groups and zero.group.mean to control it. Enabling the pre-test does not change the Dunnett.GLM results, since the state of the random number generator is restored afterwards. The pre-test is primarily a decision aid for choosing between a Poisson-based method such as CPCAT and the regression approach implemented here; within Dunnett.GLM it serves as documentation of the dispersion actually present in the data.

Details

Note that no dispersion diagnostic should be the sole basis for this decision. Any such test reduces the mean-variance relationship of a data set to a single number and, at the replication typical for ecotoxicological studies, has limited power, so that a non-significant result is weak evidence for the Poisson assumption rather than proof of it. The data should therefore always be inspected graphically as well, for example by plotting the group variances against the group means together with the identity line, or the residuals against the fitted values, and by looking at the per-group variance-to-mean ratios that Overdispersion.test reports.

The information object that is returned alongside the test results reports the distribution and the link function that were used, together with the dispersion parameter. For family = "quasipoisson" this is the estimated dispersion of the Quasi-Poisson model, for family = "nb" it is the estimated theta of the negative binomial model including its standard error, and for family = "poisson" it is noted that the dispersion is fixed at 1 by the model assumption.

Note on zero treatments and the negative binomial model: zero.treatment.action only takes effect if a treatment consists exclusively of zeros. In that situation family = "nb" cannot be combined with zero.treatment.action = "identity.link", and Dunnett.GLM stops with a message that says so. The

reason is that an identity link forces the fitted mean of such a treatment to zero. A treatment with a mean of zero also has a variance of zero and therefore carries no information on the dispersion, so the negative binomial dispersion parameter is not identifiable at that point, and its maximum likelihood estimation divides by the fitted mean. Whether that mean is represented as a small positive number, as exactly zero or as a small negative number depends on floating point rounding and differs between platforms, which would make the result machine-dependent.

Use zero.treatment.action = "log(x+1)" together with family = "nb" instead. Adding one to every count removes the degenerate mean, the log link is retained and the dispersion is estimated without difficulty. If the identity link is required, use family = "quasipoisson" or family = "poisson", neither of which estimates a dispersion parameter that depends on the fitted mean in this way. Note that the two options also differ in interpretation: with a log link the Dunnett contrasts compare the treatments with the control on a multiplicative scale, with an identity link on an additive one.

Value

R object with results and information from Dunnett.GLM calculations

References

Hothorn, L.; Kluxen, F. (2020): Statistical analysis of no observed effect concentrations or levels in eco-toxicological assays with overdispersed count endpoints. In: bioRxiv, 2020, https://doi.org/10.1101/2020.01.15.907881

Venables, W. N.; Ripley, B. D. (2002): Modern Applied Statistics with S. Fourth Edition. Springer, New York.

Examples

```
Daphnia.counts # example data provided alongside the package

# Test Dunnett.GLM with 'identity.link' option
Dunnett.GLM(groups = Daphnia.counts$Concentration,
      counts = Daphnia.counts$Number_Young,
      control.name = NULL,
      zero.treatment.action = "identity.link",
      show.output = TRUE)

# Test Dunnett.GLM with 'log(x+1)' option
Dunnett.GLM(groups = Daphnia.counts$Concentration,
     counts = Daphnia.counts$Number_Young,
     control.name = NULL,
     zero.treatment.action = "log(x+1)",
     show.output = TRUE)

# Test Dunnett.GLM using a negative binomial distribution instead of a Quasi-Poisson distribution.
# Note that the 'log(x+1)' option is used here: the example data contain a treatment consisting
# only of zeros, and the negative binomial model cannot be combined with the identity link in that
# situation (see the note on zero treatments below).
Dunnett.GLM(groups = Daphnia.counts$Concentration,
     counts = Daphnia.counts$Number_Young,
     control.name = NULL,
     zero.treatment.action = "log(x+1)",
     show.output = TRUE,
```

```
     family = "nb")

# Test Dunnett.GLM using a classical Poisson distribution, the regression counterpart of CPCAT
Dunnett.GLM(groups = Daphnia.counts$Concentration,
     counts = Daphnia.counts$Number_Young,
     control.name = NULL,
     zero.treatment.action = "log(x+1)",
     show.output = TRUE,
     family = "poisson")

# Test Dunnett.GLM and add the overdispersion pre-test to the information object
Dunnett.GLM(groups = Daphnia.counts$Concentration,
     counts = Daphnia.counts$Number_Young,
     control.name = NULL,
     zero.treatment.action = "log(x+1)",
     show.output = TRUE,
     overdispersion.test = list(bootstrap.runs = 500, use.fixed.random.seed = 123))
```

`Dunnett.GLM.bMDD` *Dunnett.GLM bootstrap MDD (bMDD)*

Description

The basic idea of the calculation of bootstrap MDD (bMDD) using the Dunnett.GLM approach is to shift the lambda parameter of Poisson distribution until there is a certain proportion of results significantly different from the control.

Usage

```
Dunnett.GLM.bMDD(
  groups,
  counts,
  control.name = NULL,
  alpha = 0.05,
  shift.step = -0.25,
  bootstrap.runs = 200,
  power = 0.8,
  max.iterations = 1000,
  use.fixed.random.seed = NULL,
  Dunnett.GLM.zero.treatment.action = "log(x+1)",
  show.progress = TRUE,
  show.results = TRUE,
  Dunnett.GLM.family = "quasipoisson"
)
```

Arguments

`groups`	Group vector

counts	Vector with count data
control.name	Character string with control group name (optional)
alpha	Significance level
shift.step	Step of shift (negative as a reduction is assumed)
bootstrap.runs	Number of bootstrap runs
power	Proportion of bootstrap.runs that return significant differences
max.iterations	Max. number of iterations to not get stuck in the while loop
use.fixed.random.seed	Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.
Dunnett.GLM.zero.treatment.action	Dunnett.GLM method to be used for treatments only containing zeros
show.progress	Show progress for each shift of lambda
show.results	Show results
Dunnett.GLM.family	Distribution family to be used for the GLM within the Dunnett.GLM method (use either "quasipoisson", "nb" or "poisson", see Dunnett.GLM). Note that this argument only affects the model that is fitted to the resampled data; the bootstrap samples themselves are always drawn from a Poisson distribution.

Value

Data frame with results from bMDD analysis

Examples

```
Daphnia.counts # example data provided alongside the package


# Test Dunnett.GLM bootstrap MDD
Dunnett.GLM.bMDD(groups = Daphnia.counts$Concentration,
 counts = Daphnia.counts$Number_Young,
 control.name = NULL,
 alpha = 0.05,
 shift.step = -1,  # Caution: big step size for testing
 bootstrap.runs = 5, # Caution: low number of bootstrap runs for testing
 power = 0.8,
 max.iterations = 1000,
 use.fixed.random.seed = 123,  #fixed seed for reproducible results
 Dunnett.GLM.zero.treatment.action = "log(x+1)",
 show.progress = TRUE,
 show.results = TRUE)

# Test Dunnett.GLM bootstrap MDD using a negative binomial distribution
Dunnett.GLM.bMDD(groups = Daphnia.counts$Concentration,
 counts = Daphnia.counts$Number_Young,
 control.name = NULL,
 alpha = 0.05,
 shift.step = -1,  # Caution: big step size for testing
```

```
 bootstrap.runs = 5, # Caution: low number of bootstrap runs for testing
 power = 0.8,
 max.iterations = 1000,
 use.fixed.random.seed = 123,  #fixed seed for reproducible results
 Dunnett.GLM.zero.treatment.action = "log(x+1)",
 show.progress = TRUE,
 show.results = TRUE,
 Dunnett.GLM.family = "nb")

# Test Dunnett.GLM bootstrap MDD using a classical Poisson distribution
Dunnett.GLM.bMDD(groups = Daphnia.counts$Concentration,
 counts = Daphnia.counts$Number_Young,
 control.name = NULL,
 alpha = 0.05,
 shift.step = -1,  # Caution: big step size for testing
 bootstrap.runs = 5, # Caution: low number of bootstrap runs for testing
 power = 0.8,
 max.iterations = 1000,
 use.fixed.random.seed = 123,  #fixed seed for reproducible results
 Dunnett.GLM.zero.treatment.action = "log(x+1)",
 show.progress = TRUE,
 show.results = TRUE,
 Dunnett.GLM.family = "poisson")
```

`Dunnett.GLM.power` *Dunnett.GLM power*

Description

The basic idea of Dunnett.GLM power calculations is to do parametric bootstrapping for each dose/concentration group and to evaluate the proportion of results significantly different from the control.

Usage

```
Dunnett.GLM.power(
  groups,
  counts,
  control.name = NULL,
  alpha = 0.05,
  bootstrap.runs = 200,
  use.fixed.random.seed = NULL,
  Dunnett.GLM.zero.treatment.action = "log(x+1)",
  show.progress = TRUE,
  show.results = TRUE,
  Dunnett.GLM.family = "quasipoisson"
)
```

Arguments

groups	Group vector
counts	Vector with count data
control.name	Character string with control group name (optional)
alpha	Significance level
bootstrap.runs	Number of bootstrap runs
use.fixed.random.seed	Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.
Dunnett.GLM.zero.treatment.action	GLM.Dunnett method to be used for treatments only containing zeros
show.progress	Show progress for each shift of lambda
show.results	Show results
Dunnett.GLM.family	Distribution family to be used for the GLM within the Dunnett.GLM method (use either "quasipoisson", "nb" or "poisson", see Dunnett.GLM). Note that this argument only affects the model that is fitted to the resampled data; the bootstrap samples themselves are always drawn from a Poisson distribution.

Value

Data frame with results from power analysis

Examples

```
Daphnia.counts # example data provided alongside the package

# Test Dunnett.GLM power
Dunnett.GLM.power(groups = Daphnia.counts$Concentration,
 counts = Daphnia.counts$Number_Young,
 control.name = NULL,
 alpha = 0.05,
 bootstrap.runs = 10, # Caution: low number of bootstrap runs for testing
 use.fixed.random.seed = 123,  #fixed seed for reproducible results
 Dunnett.GLM.zero.treatment.action = "log(x+1)",
 show.progress = TRUE,
 show.results = TRUE)


# Test Dunnett.GLM power using a negative binomial distribution
Dunnett.GLM.power(groups = Daphnia.counts$Concentration,
 counts = Daphnia.counts$Number_Young,
 control.name = NULL,
 alpha = 0.05,
 bootstrap.runs = 10, # Caution: low number of bootstrap runs for testing
 use.fixed.random.seed = 123,  #fixed seed for reproducible results
 Dunnett.GLM.zero.treatment.action = "log(x+1)",
 show.progress = TRUE,
 show.results = TRUE,
```

```
 Dunnett.GLM.family = "nb")

# Test Dunnett.GLM power using a classical Poisson distribution
Dunnett.GLM.power(groups = Daphnia.counts$Concentration,
 counts = Daphnia.counts$Number_Young,
 control.name = NULL,
 alpha = 0.05,
 bootstrap.runs = 10, # Caution: low number of bootstrap runs for testing
 use.fixed.random.seed = 123,  #fixed seed for reproducible results
 Dunnett.GLM.zero.treatment.action = "log(x+1)",
 show.progress = TRUE,
 show.results = TRUE,
 Dunnett.GLM.family = "poisson")
```

MSSR.statistic *MSSR statistic*

Description

Mean of the squared within-group Poisson-standardised residuals, T = 1/N * sum_i sum_j ((y_ij - mu_i) / sqrt(mu_i))^2. Internal helper of Overdispersion.test.

Usage

```
MSSR.statistic(group.counts, zero.group.mean = 0.5)
```

Arguments

group.counts	List with one vector of counts per treatment group
zero.group.mean	Value that replaces a group mean of zero in the denominator

Value

Numeric value of the test statistic

Multi.group.test.bMDD *Multi-group test bMDD*

Description

Idea: shift lambda of Poisson distribution until there is a certain proportion of significant results

Usage

```
Multi.group.test.bMDD(
  groups,
  counts,
  control.name = NULL,
  alpha = 0.05,
  shift.step = -0.25,
  bootstrap.runs = 200,
  power = 0.8,
  max.iterations = 1000,
  use.fixed.random.seed = NULL,
  CPCAT.bootstrap.runs = 200,
  Dunnett.GLM.zero.treatment.action = "log(x+1)",
  show.progress = TRUE,
  show.results = TRUE,
  get.effect.and.power = FALSE,
  Dunnett.GLM.family = "quasipoisson",
  test = "CPCAT"
)
```

Arguments

groups	Group vector
counts	Vector with count data
control.name	Character string with control group name (optional)
alpha	Significance level
shift.step	Step of shift (negative as a reduction is assumed)
bootstrap.runs	Number of bootstrap runs
power	Proportion of bootstrap.runs that return significant differences
max.iterations	Max. number of iterations to not get stuck in the while loop
use.fixed.random.seed	Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.
CPCAT.bootstrap.runs	Bootstrap runs within CPCAT method
Dunnett.GLM.zero.treatment.action	Dunnett.GLM method to be used for treatments only containing zeros
show.progress	Show progress for each shift of lambda
show.results	Show results
get.effect.and.power	Return effect size (percent of control) and power for each step (only for last treatment)
Dunnett.GLM.family	Distribution family to be used for the GLM within the Dunnett.GLM method (either "quasipoisson", "nb" or "poisson"). Only relevant if test = "GLM.Dunnett".
test	Either "CPCAT" or "GLM.Dunnett"

Value

Data frame with results from bMDD analysis

`Multi.group.test.power` *Multi-group test power*

Description

Idea: Do parametric bootstrapping for each group and evaluate the proportion of significant results.

Usage

```
Multi.group.test.power(
  groups,
  counts,
  control.name = NULL,
  alpha = 0.05,
  bootstrap.runs = 200,
  use.fixed.random.seed = NULL,
  CPCAT.bootstrap.runs = 200,
  Dunnett.GLM.zero.treatment.action = "log(x+1)",
  show.progress = TRUE,
  show.results = TRUE,
  Dunnett.GLM.family = "quasipoisson",
  test = "CPCAT"
)
```

Arguments

`groups`	Group vector
`counts`	Vector with count data
`control.name`	Character string with control group name (optional)
`alpha`	Significance level
`bootstrap.runs`	Number of bootstrap runs
`use.fixed.random.seed`	Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.
`CPCAT.bootstrap.runs`	Bootstrap runs within CPCAT method
`Dunnett.GLM.zero.treatment.action`	GLM.Dunnett method to be used for treatments only containing zeros
`show.progress`	Show progress for each shift of lambda
`show.results`	Show results
`Dunnett.GLM.family`	Distribution family to be used for the GLM within the Dunnett.GLM method (either "quasipoisson", "nb" or "poisson"). Only relevant if test = "GLM.Dunnett".
`test`	Either "CPCAT" or "GLM.Dunnett"

Value

Data frame with results from power analysis

`Overdispersion.test` *Overdispersion.test*

Description

Poisson-based methods for the determination of no/lowest observed (adverse) effect concentrations from count data, such as CPCAT, rely on the assumption that the variance of the counts equals their mean. Ecotoxicological count data frequently violate this assumption (overdispersion), which invalidates the subsequent inference. Overdispersion.test provides a model-agnostic pre-test that checks this assumption before a method is chosen. Model-agnostic means that no generalised linear model has to be specified and fitted beforehand; the Poisson distribution is only used as a reference framework. The test statistic is the mean of the squared within-group Poisson-standardised residuals, T = mean(z^2) with z_ij = (y_ij - mu_i) / sqrt(mu_i), which is scale-invariant and therefore comparable across designs with different mean levels. The decision threshold is not taken from an asymptotic distribution but is calibrated by Monte Carlo simulation for the design at hand, which makes the test applicable to the small sample sizes that are typical for ecotoxicological studies. The test is one-sided: only unusually large values of T lead to rejection, i.e. underdispersion is not detected. For details on the structure of the input data, please refer to the dataset 'Daphnia.counts' provided alongside this package.

Usage

```
Overdispersion.test(
  groups,
  counts,
  alpha = 0.05,
  bootstrap.runs = 5000,
  pretest.mode = "tolerant",
  tolerance.factor = 2,
  use.fixed.random.seed = NULL,
  include.zero.groups = FALSE,
  zero.group.mean = 0.5,
  show.output = TRUE
)
```

Arguments

`groups`	Group vector
`counts`	Vector with count data
`alpha`	Significance level of the one-sided test
`bootstrap.runs`	Number of Monte Carlo simulations used to calibrate the decision threshold

`pretest.mode` Reference distribution used for the calibration (use either "tolerant" or "strict"). The "strict" option calibrates against a pure Poisson distribution, i.e. any deviation from variance = mean is treated as overdispersion. The "tolerant" option calibrates against a negative binomial distribution with a tolerated variance-to-mean ratio (see tolerance.factor), i.e. mild deviations from the Poisson assumption are accepted. The tolerant option reflects that Poisson-based methods are robust against small deviations and that a threshold is needed above which they are not.

`tolerance.factor`
Tolerated variance-to-mean ratio used if pretest.mode = "tolerant". The negative binomial size parameter is set to k = mu / (tolerance.factor - 1), which yields a variance of mu * tolerance.factor, i.e. the tolerance factor is exactly the tolerated variance-to-mean ratio. Must be greater than 1. Ignored if pretest.mode = "strict".

`use.fixed.random.seed`
Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.

`include.zero.groups`
Include treatment groups whose counts are all zero. Such groups carry no information on dispersion and are excluded by default.

`zero.group.mean`
Value that replaces a group mean of zero in the denominator of the standardised residuals. Only used if include.zero.groups = TRUE.

`show.output` Show/hide output

Details

The observed statistic T is compared with a critical value c_alpha that is obtained as the (1 - alpha) quantile of the Monte Carlo null distribution of T. Overdispersion is reported if T > c_alpha. In addition, a one-sided Monte Carlo p-value is calculated as (1 + number of simulated values >= T) / (1 + bootstrap.runs).

Note on the interpretation of T: under a Poisson distribution with KNOWN group means the expected value of z^2 is 1. Since the group means are estimated from the same data, the expected value of T is (N - G) / N, with N being the number of observations and G the number of groups, i.e. it is smaller than 1 and depends on the design (for example 0.833 for six replicates per group). This does not affect the validity of the test, because the Monte Carlo calibration uses exactly the same estimator and the bias therefore cancels out. To provide a directly interpretable quantity, the results additionally report the bias-corrected dispersion ratio phi = T * N / (N - G), which has an expected value of 1 under a Poisson distribution and can be read as a pooled variance-to-mean ratio. Since phi is a strictly monotone transformation of T, the test decision is identical in both scales.

The decision of this test should not be used as the only criterion for the choice of a method. The statistic reduces the mean-variance relationship of a whole data set to a single number, and at the replication typical for ecotoxicological studies the test has limited power, so that a non-significant result is weak evidence for the Poisson assumption rather than proof of it. The group statistics returned alongside the test report the mean, the variance and the variance-to-mean ratio of every treatment, and these should be inspected as well, together with a graphical check such as a plot of the group variances against the group means with the identity line added, or of the residuals against the fitted values. A single treatment with a markedly deviating variance-to-mean ratio is visible there but may well leave the pooled statistic non-significant.

Groups containing fewer than two observations are excluded, since a within-group dispersion cannot be evaluated for them.

Value

R object with results, group statistics and information from the overdispersion test

References

Daniels, B.; Kotschik, P.; Walter-Rohde, S.; Graeff, T. (2026): When Counts Deviate: Robust Overdispersion Diagnostics for Ecotoxicity Data. Poster presented at the 36th SETAC Europe Annual Meeting, 17-21 May 2026, Maastricht, The Netherlands.

Cameron, A. C.; Trivedi, P. K. (1990): Regression-based tests for overdispersion in the Poisson model. In: Journal of Econometrics, 1990, 46(3), p. 347-364, https://doi.org/10.1016/0304-4076(90)90014-K

Lehmann, R.; Bachmann, J.; Maletzki, D.; Polleichtner, C.; Ratte, H.; Ratte, M. (2016): A new approach to overcome shortcomings with multiple testing of reproduction data in ecotoxicology. In: Stochastic Environmental Research and Risk Assessment, 2016, 30(3), p. 871-882, https://doi.org/10.1007/s00477-015-1079-4

Examples

```
Daphnia.counts # example data provided alongside the package

# Test for overdispersion, tolerating a variance-to-mean ratio of up to 2
Overdispersion.test(groups = Daphnia.counts$Concentration,
  counts = Daphnia.counts$Number_Young,
  alpha = 0.05,
  bootstrap.runs = 500, # Caution: low number of simulations for testing
  pretest.mode = "tolerant",
  tolerance.factor = 2,
  use.fixed.random.seed = 123,  #fixed seed for reproducible results
  show.output = TRUE)

# Test against a pure Poisson distribution, i.e. without tolerating any deviation
Overdispersion.test(groups = Daphnia.counts$Concentration,
  counts = Daphnia.counts$Number_Young,
  alpha = 0.05,
  bootstrap.runs = 500, # Caution: low number of simulations for testing
  pretest.mode = "strict",
  use.fixed.random.seed = 123,  #fixed seed for reproducible results
  show.output = TRUE)
```

`run.overdispersion.diagnostic` *Overdispersion diagnostic within a test function*

Description

Runs Overdispersion.test from within CPCAT or Dunnett.GLM and returns a single line of text for their information object. The state of the random number generator is saved and restored, so that enabling the diagnostic does not change the results of the calling function. Internal helper.

Usage

```
run.overdispersion.diagnostic(dat, settings)
```

Arguments

`dat`	Data frame with the columns Counts and Groups
`settings`	Either NULL/FALSE (diagnostic switched off), TRUE (default settings) or a named list with arguments that are passed on to Overdispersion.test

Value

Character string to be added to the information object, or NULL if switched off

Index

∗ **datasets**
CPFISH.contingency.table, 12
Daphnia.counts, 14

calibrate.c.alpha, 2
CP.hypotheses, 3
CPCAT, 4
CPCAT.bMDD, 6
CPCAT.Poisson.sub.test, 7
CPCAT.Poisson.test, 8
CPCAT.power, 8
CPFISH, 10
CPFISH.bMDD, 11
CPFISH.contingency.table, 12
CPFISH.power, 13

Daphnia.counts, 14
Dunnett.GLM, 15, *19*, *21*
Dunnett.GLM.bMDD, 18
Dunnett.GLM.power, 20

glm.nb, *16*

MSSR.statistic, 22
Multi.group.test.bMDD, 22
Multi.group.test.power, 24

Overdispersion.test, *5*, *16*, 25

run.overdispersion.diagnostic, 27