

# Package ‘qountstat’

July 22, 2025

**Type** Package

**Title** Statistical Analysis of Count Data and Quantal Data

**Version** 0.1.1

**Maintainer** Benjamin Daniels <Benjamin.Daniels@uba.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](https://doi.org/10.1007/s00477-015-1079-4)>), as well as an implementation of a ‘‘Dunnett GLM’’ approach using a Quasi-Poisson regression (Hothorn, L, Kluxen, F (2020) <[doi:10.1101/2020.01.15.907881](https://doi.org/10.1101/2020.01.15.907881)>).

For the analysis of quantal data an implementation of the Closure Principle Fisher–Freeman–Halton test (‘‘CPFISH’’) is pro-

vided (Lehmann, R et al. (2018) <[doi:10.1007/s00477-017-1392-1](https://doi.org/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.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** multcomp

**Suggests** knitr, rmarkdown, testthat

**RoxygenNote** 7.3.2

**Depends** R (>= 2.10)

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2025-02-21 11:30:25 UTC

## Contents

CP.hypotheses . . . . .	2
CPCAT . . . . .	3
CPCAT.bMDD . . . . .	4
CPCAT.Poisson.sub.test . . . . .	6
CPCAT.Poisson.test . . . . .	6
CPCAT.power . . . . .	7
CPFISH . . . . .	8
CPFISH.bMDD . . . . .	9
CPFISH.contingency.table . . . . .	10
CPFISH.power . . . . .	11
Daphnia.counts . . . . .	12
Dunnett.GLM . . . . .	13
Dunnett.GLM.bMDD . . . . .	14
Dunnett.GLM.power . . . . .	16
Multi.group.test.bMDD . . . . .	17
Multi.group.test.power . . . . .	18
<b>Index</b>	<b>20</b>

---

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 = 5,
  use.fixed.random.seed = NULL,
  get.contrasts.and.p.values = FALSE,
  show.output = TRUE
)
```

**Arguments**

<code>groups</code>	Group vector
<code>counts</code>	Vector with count data
<code>control.name</code>	Character string with control group name (optional)
<code>bootstrap.runs</code>	Number of bootstrap runs
<code>hampel.threshold</code>	Threshold for Hampel identifier (measure for over-/underdispersion)
<code>use.fixed.random.seed</code>	Use fixed seed, e.g. 123, for reproducible results. If NULL no seed is set.
<code>get.contrasts.and.p.values</code>	Get each row of the contrast matrices evaluated
<code>show.output</code>	Show/hide output

**Value**

R object with results and information from CPCAT calculations

**References**

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)
```

---

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	<i>CPFISH power</i>
--------------	---------------------

---

**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**

<code>contingency.table</code>	Matrix with observed data (e.g. survival counts, survival must be in first row)
<code>control.name</code>	Character string with control group name (optional)
<code>alpha</code>	Significance level
<code>bootstrap.runs</code>	Number of bootstrap runs (draw Poisson data n times)
<code>simulate.p.value</code>	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	<i>Daphnia.counts is a dataset to showcase analyses with CPCAT and Dunnett.GLM</i>
----------------	--

---

**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. 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
)
```

**Arguments**

<code>groups</code>	Group vector
<code>counts</code>	Vector with count data
<code>control.name</code>	Character string with control group name (optional)
<code>zero.treatment.action</code>	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.
<code>show.output</code>	Show/hide output

**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>

## 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)
```

---

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
)
```

**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

**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)
```

---

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
)
```

### 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

### 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)
```

---

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,
  use.CMP.distribution = FALSE,
  CMP.dispersion.factor = 1,
  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)
use.CMP.distribution	Use Conway-Maxwell-Poisson distribution for sampling
CMP.dispersion.factor	Dispersion parameter phi has to be sqrt(factor) to scale the variance by this factor
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,  
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
test	Either "CPCAT" or "GLM.Dunnett"

### Value

Data frame with results from power analysis

# Index

## \* datasets

CPFISH.contingency.table, [10](#)

Daphnia.counts, [12](#)

CP.hypotheses, [2](#)

CPCAT, [3](#)

CPCAT.bMDD, [4](#)

CPCAT.Poisson.sub.test, [6](#)

CPCAT.Poisson.test, [6](#)

CPCAT.power, [7](#)

CPFISH, [8](#)

CPFISH.bMDD, [9](#)

CPFISH.contingency.table, [10](#)

CPFISH.power, [11](#)

Daphnia.counts, [12](#)

Dunnett.GLM, [13](#)

Dunnett.GLM.bMDD, [14](#)

Dunnett.GLM.power, [16](#)

Multi.group.test.bMDD, [17](#)

Multi.group.test.power, [18](#)