Package 'emon'

May 8, 2026

Type Package

Title Tools for Environmental and Ecological Survey Design

Version 1.3.2

Date 2017-03-03

Author Jon Barry and David Maxwell

Maintainer Jon Barry <jon.barry@cefas.co.uk>

Imports mgcv, MASS

Description Statistical tools for environmental and ecological surveys.
Simulation-based power and precision analysis; detection probabilities from
different survey designs; visual fast count estimation.

License GPL-3

Repository CRAN

Repository/R-Forge/Project emon

Repository/R-Forge/Revision 14

Repository/R-Forge/DateTimeStamp 2017-03-09 11:39:30

Date/Publication 2017-03-09 14:50:09

NeedsCompilation no

Contents

`emon-package` *Tools for environmental and ecological survey design and analysis*

Description

This package gives seven tools for designing and analysing ecological and environmental surveys. The tools are mainly designed for marine and benthic ecology applications, but they could easily be adopted for terrestrial ecology. Three of the tools give statistical power for specific survey designs (`power.BACI`, `power.groups` and `power.trend`). The fourth tool (`precision`) calculates the sample size needed to achieve specified precision for estimating the mean of some desired statistic together with the precision obtained for given `n`.

The other three tools are for more specialised applications. These are: the generalised visual fast count estimator for underwater video surveys (`GVFCMOM`); an estimate of the empirical semi-variogram for examining spatial correlation between stations (`svariog`); and detection probability for three spatial sampling designs (`detect` and `detect.prop`).

Details

Package:	emon
Type:	Package
Version:	1.3.2
Date:	2017-03-03
License:	GPL-3

The seven tools in this package are as follows:

Power for BACI designs (`power.BACI`, `generate.trend`, `addnoise`, `mannkendall`, `mannkendall.stat`, `permute.BACI`).

Power for comparing two groups (`power.groups`, `permute.groups`, `size2.samevar`).

Power for detecting trends (power.trend, generate.trend, addnoise).

Precision for estimating a mean (precision).

Sample sizes and probabilities for patch detection with different spatial sampling patterns (detect, detect.prop, fS.detect, fT.detect).

Semi-variogram function for investigating spatial dependency (svariog).

Method of moments estimator for Generalised Visual Fast Count estimation for video surveys (GVFCMOM, GVFC, expected.pois, expected.nb, mom.min.pois, mom.min.nb).

The help functions for the individual functions describe the methods used. However, perhaps the unique feature of the power functions in emon is that the statistical power is calculated by simulation. This has the disadvantage of increased computing time; however, the advantage is that the power calculations does not rely on the assumptions behind many of the theoretical results. The simulation method also means that power can be calculated for a range of data distributions and for a variety of statistical tests that might be used to evaluate p-values.

Author(s)

Jon Barry and David Maxwell

Maintainer: Jon Barry: Jon.Barry@cefas.co.uk

See Also

power.BACI, power.groups, power.trend, precision. detect, svariog, GVFCMOM

---

addnoise *Creates random errors for use in power.trend.*

Description

This function is used within power.trend. Distribution of noise can be either Normal, Poisson or Negative Binomial. The mean values are entered as a parameter (possibly generated by function generate.trend). Other parameters (sd for Normal, nbsize for Negative Binomial) need to be given. Values in meanvalues are used as the mean values for one of the three specified distributions and then a random allocation is made for each of the nobs values, where nobs is the length of meanvalues.

Usage

```
addnoise(meanvalues, reps, distribution, sd, nbsize, randeffect, randeffect.sd)
```

Arguments

meanvalues	Vector of mean values
reps	Number of replicates per time point
distribution	Character string which must be one of Normal (default), Poisson or Negbin
sd	Standard Deviation for Normal distribution

nbsize	Size parameter for Negative Binomial distribution
randeffect	Not working yet
randeffect.sd	Not working yet

Value

Output is a vector of the same length as meanvalues.

Author(s)

David Maxwell: David.Maxwell@cefas.co.uk

See Also

generate.trend, power.trend

---

detect *Probability of circular patch detection*

Description

The function can calculate the probability of detection of a circular patch of specified radius for a specified density of points; the density needed to achieve a specified probability of detection; or the radius of the patch that will be detected with specified probability and sampling density.This is done for random, square lattice, and triangular lattice spatial sampling designs.

Usage

```
detect(method, statistic, area=NA, radius=NA, pdetect=NA, ssize=NA)
```

Arguments

method	Defines the spatial sampling design to be used. The values can be "R" (random), "S" (square lattice) or "T" (triangular lattice). See Barry and Nicholson (1993) for details and formulae for the probabilities of detection for the square lattice and triangular lattice designs. For the random design, prob(detect)=1 - (1 - a/A)^N, where a is the patch area and A is the survey area. This gives similar answers to the formula in Barry and Nicholson, but is exact for fixed sample size.
statistic	Describes what aspect of design you want calculated. The choices are "P" (probability detection); "N" (sample size) or "R" (patch radius).
area	The survey area (same units as distance and radius).
radius	Patch radius. Not needed if statistic="R".
pdetect	Probability detection. Not needed if statistic="P".
ssize	Sample size. Not needed if statistic="N".

Details

The basic idea is that you wish to conduct a survey in an area `area` to detect some object (patch) of interest. This could be a cockle patch, an area of reef or an archaeological deposit. This function asssumes that the object is circular with radius `radius`. You have three choices of sampling deign to use: spatial, square lattice and triangular lattice. In terms of patch detection, for a given sample size, the triangular design gives the highest probability - because its points are equi-distant apart.

The simplest application of this function is to assess the patch detection probability for a particular design. This is obtained using the `statistic="P"` option. However, the problem can be turned around and this function used to calculate the sample size needed to obatain a specific patch detection probability (`statistic="N"`) or the radius of the patch that would be detected with some desired probability (`statistic="R"`). This last scenario might be useful if there was some particular size of patch that you wanted to be sure (say, 90 percent) of detecting.

Value

`prob`	Probability of patch detection
`ssize`	Sample size
`rad`	Patch radius
`sep`	Separation distance (for square and triangular lattice designs)

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

References

Barry J and Nicholson M D (1993) Measuring the probabilities of patch detection for four spatial sampling schemes. Journal of Applied Statistics, 20, 353-362.

Examples

```
detect(method='R', statistic='P', area=100, radius=2, ssize=15)$prob
detect(method='R', statistic='N', area=100, radius=2, pdetect=0.95)$ssize
detect(method='R', statistic='R', area=100, pdetect=0.95, ssize=15)$rad

detect(method='S', statistic='P', area=100, radius=1.4, ssize=15)
detect(method='S', statistic='N', area=100, radius=1.4, pdetect=0.6)

# Plot patch detection as a function of radius
square = rep(0,200); rand = square; triang = rand
radius = seq(0.01, 2, 0.01)

for (j in 1:200) {
rand[j] = detect(method='R', statistic='P', area=100, radius=radius[j], ssize=15)$p
square[j] = detect(method='S', statistic='P', area=100, radius=radius[j], ssize=15)$p
triang[j] = detect(method='T', statistic='P', area=100, radius=radius[j], ssize=15)$p
}

plot(radius, rand, ylim=c(0,1), xlab='Patch radius', ylab='Probability detection', type='l')
```

```
lines(radius, square, col=2, lty=2)
lines(radius, triang, col=3, lty=3)
legend('topleft', legend=c('Random', 'Square', 'Triangular'), col=c(1,2,3), lty=c(1,2,3))
```

`detect.prop` *Probability of detecting a feature that covers a proportion* `theta` *of the survey area.*

Description

The function can calculate the probability of a feature that occupies a proportion `theta` of the sampling area and where the sampling point density of the survey is specified; the sampling point density needed to achieve a specified probability of detection, where `theta` is also specified ; or the value of `theta` that will be detected with specified probability and sampling density.Unless the feature is made of a large number of random segments (see below for how to deal with this situation), these methods apply only when the pattern of points in the sampling deisgn is random.

Usage

```
detect.prop(statistic, theta=NA, pdetect=NA, ssize=NA)
```

Arguments

`statistic`	Describes what aspect of design you want calculated. The choices are `"P"` (probability detection); `"N"` (sample size) or `"F"` (feature proportion).
`theta`	Feature proportion. Not needed if `statistic="F"`.
`pdetect`	Probability detection. Not needed if `statistic="P"`.
`ssize`	Sample size. Not needed if `statistic="N"`.

Details

The probability of detection is `p = 1 - (1 - theta)^{N}`. Formulae for `theta` and `N` are readily obtained from this formula. If the spatial pattern of the feature consists of lots of small, random uniformly distributed fragments, then we can redefine `theta = Na/A` where `a` is the area of the sampling unit and `A` is the sampling area.In this situation, the probability of patch detection applies no matter what the spatial pattern of points in the sampling design. Unlike `detect`, `detect.prop` works for vectors - so long as the input vectors are of the same length.

Value

`prob`	Probability of detection
`ssize`	Sample size
`prop`	Feature proportion

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

Examples

```
detect.prop(statistic='P', theta=0.02, ssize=80)
detect.prop(statistic='N', theta=0.02, pdetect=0.9)
detect.prop(statistic='F', pdetect=0.9, ssize=80)
```

`expected.nb` *Expected value of Visual Fast Count Estimator assuming Negative Binomial distribution for counts*

Description

The function is used to obtain the method of moments estimator within the function GVFCMOM.

Calculates the expected value of the Visual Fast Count method. The function assumes that the count per segment is Negative Binomial with mean `m/s` and size `k`, and that segment counts are independent. The expected value is also a function of the number of positives `d` before segment counting is stopped.

Usage

```
expected.nb(k, m, s, d)
```

Arguments

`k`	Size parameter of the Negative Binomial distribution
`m`	The mean of the Negative Binomial distribution per transect
`s`	The number of segments per transect
`d`	The number of positive counts before segment counting is stopped

Value

The expected value of the Visual Fast Count estimator

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

See Also

`expected.pois`, GVFCMOM

expected.pois — *Expected value of Visual Fast Count Estimator assuming Poisson distribution for counts*

Description

The function is used to obtain the method of moments estimator within the function GVFCMOM. Calculates the expected value of the Visual Fast Count method. The function assumes that the count per segment is Poisson with mean m/s and that segment counts are independent. The expected value is also a function of the number of positives d before segment counting is stopped.

Usage

```
expected.pois(m, s, d)
```

Arguments

m	The underlying mean of the Poisson process per transect
s	The number of segments per transect
d	The numbner of positive counts before segment counting is stopped

Value

The expected value of the Visual Fast Count estimator

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

See Also

expected.nb, GVFCMOM

fS.detect — *Used in the function detect*

Description

Used in the function detect for calculating the sample size for a square lattice design. It is used by the optimize function.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

See Also

detect

fT.detect *Used in function detect*

Description

Used in the function detect for calculating the sample size for a triangular lattice design. It is used by the optimize function.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

See Also

detect

generate.trend *Generates a set of mean values.*

Description

This function is used to generate mean value scenarios for use in power.trend.

Usage

```
generate.trend(nyears, mu1=0, change, change.type="A", type = c("linear", "incident",
       "step", "updown"), changeyear, symmetric=F)
```

Arguments

nyears	Defines the number of time points for X axis
mu1	The mean Y value for the first time point
change	Difference between mu1 and largest (or smallest) mu_i. Can be negative for a decreasing trend.
change.type	Whether the parameter change represents an additive ("A") or percentage ("M") change.
type	Type of trend to assess the power against. Can be any of "linear", "incident", "step" or "updown". See examples below for more details.
changeyear	Year in which change in gradient occurs, for options 'incident' or 'updown'.
symmetric	If symmetric=T, nyears is even and changeyear = nyears / 2 then type='updown' generates a pattern where the two middle years are at level k.

Details

Assumes that surveys take place in years 1 to `nyears` (or could be any other equally spaced unit). Generates a set of mean values (or signal) for a specified trend over that time period. The approach is based on Fryer and Nicholson (1993, 1999). For `type="linear"`, the slope is generated by `b=k/(nyears-1)`. If `type="updown"`, the slope until `changeyear` is generated by `b=k/(changeyears-1)`. After `changeyear`, the slope is -b (except for the special case outlined by the `symmetric` parameter above, where `changeyear` and `changeyear+1` are k and then the slope continues as -b).

Value

Generates a data frame where the first column ($i) is year and the second column ($mu) is the mean value.

Author(s)

David Maxwell: David.Maxwell@cefas.co.uk

References

Fryer RJ & Nicholson MD (1993) The power of a contaminant monitoring programme to detect linear trends and incidents. ICES Journal of Marine Science, 50, 161-168.

Fryer & Nicholson 1999 Using smoothers for comprehensive assessments of contaminant time series in marine biota. ICES Journal of Marine Science, 56, 779-790.

See Also

`power.trend`, `addnoise`

Examples

```
lin0 = generate.trend(nyears=10, change=0, type="linear")
lin5 = generate.trend(nyears=10, change=5, type="linear")
inc5 = generate.trend(nyears=10, change=5, type="incident", changeyear=6)
step5 = generate.trend(nyears=10, change=5, type="step", changeyear=6)
updeven = generate.trend(nyears=10, change=5, type="updown", changeyear=5, symmetric=TRUE)
updodd = generate.trend(nyears=9, change=5, type="updown", changeyear=3)

par(mfrow=c(2,3))
plot(lin0$i, lin0$mu, type="o", pch=16, xlab='Time', ylab='Y')
plot(lin5$i, lin5$mu, type="o", pch=16, xlab='Time', ylab='Y')
plot(inc5$i, inc5$mu, type="o", pch=16, xlab='Time', ylab='Y')
plot(step5$i, step5$mu, type="o", pch=16, xlab='Time', ylab='Y')
plot(updeven$i, updeven$mu, type="o", pch=16, xlab='Time', ylab='Y')
plot(updodd$i, updodd$mu, type="o", pch=16, xlab='Time', ylab='Y')
```

GVFC *Calculates the raw Visual Fast Count (VFC) estimator of the mean abundance per transect*

Description

The function considers the counts per segment and uses them sequentially until d positive counts are obtained or until all s segments have been considered. If we assume that u counts are used (of which some may be zero) then the visual fast count estimator is the mean over the u counts multiplied by s. This function is used by GVFCMOM to obtain the method of moments VFC estimator - which has reduced bias compared to the raw VFC estimator.

Usage

```
GVFC(counts, s, d)
```

Arguments

counts	Vector of length s that contains a count for each segment
s	Number of segments
d	Number of positive segment counts needed before counting stops

Value

The raw VFC estimate of the segment mean

Author(s)

Jon Barry: jon.barry@cefas.co.uk

References

Barry J, Eggleton J, Ware S and Curtis M (2015) Generalizing Visual Fast Count Estimators for Underwater Video Surveys. Ecosphere. http://www.esajournals.org/doi/full/10.1890/ES15-00093.1

See Also

GVFCMOM

GVFCMOM *Function to calculate the method of moments visual fast count estimator*

Description

The function takes data in the form of counts per segment along a transect and uses the raw Generalised Visual Fast Count estimator (as calculated by GVFCMOM and its expectation (as calculated by expected.pois for Poisson or expected.nb for negative binomial) to calculate a method of moments estimator. This effectively, adjusts the biased raw GVFC estimate. The function allows counts to have either a Poisson or a Negative Binomial distribution. The method is a generalisation of the methods in Barry and Coggan (2010).

Usage

```
GVFCMOM(counts, s, d, method, k=1, lowint=0, highint=100)
```

Arguments

counts	Vector of length s that contains a count for each segment
s	Number of segments
d	Number of positive segment counts needed before counting stops
method	Whether Poisson ("pois") or Negative Binomial Distribution ("nb") is assumed
k	Size parameter of the Negative Binomial distribution
lowint	Minimum value for MOM estimate (default=0)
highint	Maximum value for MOM estimate (default=100)

Value

The method of moments estimate for the transect is returned

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

References

Barry J, Eggleton J, Ware S and Curtis M (2015) Generalizing Visual Fast Count Estimators for Underwater Video Surveys. Ecosphere. http://www.esajournals.org/doi/full/10.1890/ES15-00093.1

See Also

GVFC, expected.pois, expected.nb

Examples

```
counts = c(0, 0, 0, 0, 1, 1, 1, 2, 1)
GVFCMOM(counts, s=9, d=2, method='nb', lowint=0, highint=100)
GVFCMOM(counts, s=9, d=1, method='nb', lowint=0, highint=100)
GVFCMOM(counts, s=9, d=1, method='pois', lowint=0, highint=100)
```

---

is.wholenumber *To check whether an argument is an integer*

Description

Used in error checking to ascertain whether a function argument is an integer.

Usage

```
is.wholenumber(x, tol = .Machine$double.eps^0.5)
```

Arguments

x	Number to be checked.
tol	If x is closer to an integer than this, then it passes.

Value

Vector of logical values if the corresponding input values is an integer or not.

References

is.wholenumber is taken from the is.integer help file.

Examples

```
is.wholenumber( seq(1, 5, by = 0.5) ) #-->  TRUE FALSE TRUE ...
```

---

mannkendall *Mann-kendall test for trend*

Description

Calculates the Mann-Kendall statistic for monotonic trend and also the p-value against the null hypothesis of no trend. Unlike the function MannKendall, works for repeat values of time.

Usage

```
mannkendall(time, Y, nsims.mk=999)
```

Arguments

time	Vector of values which define the direction of the trend.
Y	Vector of values for which you want to determine the trend.
nsims.mk	Number of replicate permuations to calculate the p-value. Default=999.

Details

Error checks of parameters not included so as not to slow down mannkendall. The statistic is calculated by considering each case j and considering the subset of observations that have time greater than time[j]. The Mann Kendall statistic is the number of observations in this subset for which Y > Y[j] minus the number for which Y < Y[j]. The statistic is summed over all j. The p-value is calculated by nreps random permutations of the Y values.

Value

mann	Mann-Kendall statistic of observed data, as calculated by mannkendall.stat.
pvalue	P-value assuming a null hypothesis of no trend and two-way alternative hypothesis.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

References

Mann, H.B. (1945), Nonparametric tests against trend, Econometrica, 13, 245-259.

Kendall, M.G. 1975. Rank Correlation Methods, 4th edition, Charles Griffin, London.

See Also

mannkendall.stat, power.trend, MannKendall

Examples

```
x = rep(1:10,rep(2,10))
y = rnorm(20, 5, 2)
mannkendall(x, y)
```

`mannkendall.stat` *Mann-Kendall statistic.*

Description

Calculates the Mann-Kendall statistic for monotonic trend. Unlike the function `MannKendall`, works for repeat values of time. Used in function `mannkendall`, which also calculates the p-value by simulation.

Usage

```
mannkendall.stat(time, Y)
```

Arguments

`time`	Vector of values which define the direction of the trend.
`Y`	Vector of values for which you want to determine the trend.

Details

The statistic is calculated by considering each case `j` and considering the subset of observations that have time greater than `time[j]`. The Mann Kendall statistic is the number of observations in this subset for which `Y > Y[j]` minus the number for which `Y < Y[j]`. The statistic is summed over all `j`. The p-value is calculated by `nreps` random permutations of the Y values.

Value

Mann-Kendall statistic

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

References

Mann, H.B. (1945), Nonparametric tests against trend, Econometrica, 13, 245-259. Kendall, M.G. 1975. Rank Correlation Methods, 4th edition, Charles Griffin, London.

See Also

`mannkendall`, `power.trend`, `MannKendall`

mom.min.nb — *Minimising function for VFC MOM estimator assuming Negative Binomial counts.*

Description

Used by the optimize function to obtain the method of moments estimator within the function GVFCMOM.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

See Also

optimize, GVFCMOM

mom.min.pois — *Minimising function for VFC MOM estimator assuming Poisson counts.*

Description

Used by the optimize function to obtain the method of moments estimator within the function GVFCMOM.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

See Also

optimize, GVFCMOM

n.min — *Minimising function used in* precision.

Description

Used by the optimize function to obtain the correct sample size within the function precision given that the t-distributions is also a function of sample size.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

See Also

optimize, precision

permute.BACI — *Does non-parametric randomisation test for the interaction term in a BACI design.*

Description

We have control and treatment data from time 1 in a BACI design, plus control and treatment data from time 2. The interaction the amount that the difference in the control and treatment meansis different between times 1 and 2.

Usage

```
permute.BACI(t1, c1, t2, c2, nreps=999)
```

Arguments

t1	Data vector for the treatment at time 1
c1	Data vector for the control at time 1
t2	Data vector for the treatment at time 2
c2	Data vector for the control at time 2
nreps	Number of replications used in the randomisation and generation of the p-value. Default is nreps=999

Details

There are several permutation that can be used to generate the null distribution for the interaction (see Manly, 2006 and Anderson and Terr Braak, 2003). The method used here is to do a complete randomisation of the raw data.

The p-value is calculated as suggested by Manly (2006).

Value

The p-value is returned as `$p.value`

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

References

Manly BFJ (2006) Randomization, Bootstrap And Monte Carlo Methods in Biology: 3rd edition. Chapman and Hall.

Anderson, M.J. and Ter Braak, C.J.F. (2003). Permutation tests for multi-factorial analysis of variance. Journal of Computation and Simulation, 73, 85-113.

See Also

`power.BACI`, `permute.groups`

`permute.groups` *Does randomisation test for the difference in means of two vectors* `v1` *and* `v2`.

Description

Does randomisation test for the difference in means mu1, mu2 of two vectors `v1` and `v2`. Can do one or two sided tests.

Usage

```
permute.groups(v1, v2, alternative, nreps)
```

Arguments

`v1`	Data vector for variable 1
`v2`	Data vector for variable 2
`alternative`	A character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` (`mu1>mu2`) or `"less"`. You can specify just the initial letter.
`nreps`	Number of replications used in the randomisation and generation of the p-value. Default is `nreps=999`

Details

Under the null hypothesis that `mu1=mu2`, the labelling of the `n1+n2` observations is unimportant. Therefore, we can generate the null distribution for the test statistic `m1-m2` or `|m1-m2|` depending on whether a one or two sided test is required) by randomly permuting the treatment labels nreps times and calculating the test statistic each time. The p-value is calculated as suggested by Manly (2006).

Value

The p-value is returned as `$p.value`

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

References

Manly BFJ (2006) Randomization, Bootstrap And Monte Carlo Methods in Biology: 3rd edition. Chapman and Hall.

See Also

`power.groups`, `permute.BACI`

Examples

```
set.seed(5)
v1 = rnorm(27,10,2); v2=rnorm(25,11,2)
permute.groups(v1, v2, alternative="two")
permute.groups(v1, v2, alt="l")
```

`power.BACI` *Calculates power for a Before and After Control Impact (BACI) design.*

Description

BACI designs are commonly used in environmental monitoring. They are relevant where you want to measure the effect of an impact (e.g. the effect on benthic ecology of dredging in an area). Observations for treatment and control areas are measured BEFORE and after the impact. This function allows you to examine the power of particular BACI designs to detect differences between the control and the treatment.

Usage

```
power.BACI(change, change.type="M", nt, nc, parst, parsc,
         distribution, test="P", alpha=0.05, nsims=1000)
```

Arguments

`change`	AFTER treatment mean minus BEFORE treatment mean or percentage change of AFTER treatment mean relative to BEFORE treatment mean (depending on value of `change.type`).
`change.type`	Whether the parameter `change` represents an additive ("A") or percentage ("M") change.

nt	Vector of sample sizes for treatment group. Must be of same dimension as nc.
nc	Vector of sample sizes for control group. Must be of same dimension as nt.
parst	Parameters for the treatment data. If distribution="Normal", parst[1] contains the BEFORE mean and parst[2] contains the BEFORE standard deviation. If distribution="Poisson", parst[1] contains the BEFORE mean. If distribution="Lognormal", parst[1] contains the BEFORE mean of the natural log data and parst[2] contains the BEFORE standard deviation of the log data. If distribution="Negbin", parst[1] contains the BEFORE mean, parst[2] contains the BEFORE size, and parst[3] is the AFTER size.
parsc	Parameters for the control data. If distribution="Normal", parsc[1] contains the BEFORE mean and parsc[2] contains the BEFORE standard deviation. If distribution="Poisson", parsc[1] contains the BEFORE mean. If distribution="Lognormal", parsc[1] contains the BEFORE mean of the natural log data and parsc[2] contains the BEFORE standard deviation of the log data. If distribution="Negbin", parsc[1] contains the BEFORE mean, and parsc[2] contains the BEFORE size.
distribution	The statistical distribution for the two groups. Can be either: "Normal", "Poisson", "Lognormal" or "Negbin".
test	The statistical test used to compare the interaction between the control and treatment means at the BEFORE and AFTER sampling occasions. If test="NP", then the test will be a non-parametric randomisation test, in the spirit of Manly (1997), using the function permute.BACI. If test="P", then parametric tests are made to compare the treatment (i.e. a factor indicating whether an observation is from the treatmment or the control) by time (i.e. a factor indicating whether observations are BEFORE or AFTER) interaction. If distribution="Normal" then this is calculated from the usual Analysis of Variance. The same method is used (but on the log data) if distribution="Lognormal". When distribution="Poisson" or distribution="Negbin", interactions from analysis of deviance tables are used to measure the interaction. The p-value is calculated by assuming that this interaction deviance has a chi-squared distribution on 1 df. For the Negative Binomial distribution, terms in the analysis of deviance table use the same value for the size parameter as that estimated from the null model.
alpha	If the p-value for the interaction is less than alpha, a change is deemed to have been detected. Used to assess power from the nreps simulations.
nsims	Number of repeat simulations used to calculate the power. Default is 1000.

Details

BACI designs are relevant where you want to measure the effect of an impact (e.g. the effect on benthic ecology of dredging in an area). You take a number of samples both in (treatment) and outside (control) the affected area BEFORE the impact. Those in the control area should be as similar to the treatment area as possible in terms of benthic ecology. You then sample the areas again AFTER the impact has taken place. If there is an interaction for the 2x2 crossed design then

there is an effect of the impact. That is, if the control area has changed differently to the treatment area.

This function allows you to examine the power of particular BACI designs. The distribution of the measure being used can be Normal, Poisson, Negative Binomial or Lognormal.

It is also assumed that the sample sizes before and after the impact are the same, although the sample size in the treatment area can be different to the control area. Thus, if 10 and 8 samples are taken in the treatment and control areas before the impact, then 10 and 8 samples are assumed to be taken after the impact.

For the Normal, Poisson and Negative Binomial distributions, the parameter change is simply the percentage or additive change of the treatment mean from the BEFORE to the AFTER sampling occasions. For the Lognormal distribution, the percentage change is relative to the BEFORE treatment mean on the non-log scale. The BEFORE treatment mean is estimated from the mean of the log data (`parst[1]`), the standard deviation of the log data (`parst[2]`) and the proposed sampling size `nt`.

The estimator used is the one proposed by Shen (2006), which did better in terms of mean squared error than both the sample mean on the non-log scale and the maximum likelihood estimators. This is given by `mean.before = exp(parst[1] + (nt-1)*ss / (2*(nt+4)*(nt-1) + 3*ss))`, where `ss = (nt-1)*parst[2]**2`.

The Negative Binomial distribution option (`parst[3]` allows the user to specify the size parameter of the AFTER treatment distribution. One possibility is to keep the size the same for both the BEFORE and AFTER distributions. However, because the mean changes and because the variance V = mu+mu^2/size, this means that V will be different for the BEFORE and AFTER distributions. If you want to keep the variance the same, you can use the function `size2.samevar`.

Several powers can be calculated per call to this function by specifying more than one values for the sample sizes `nt` and `nc`.

Value

`power`	The estimated power for the design.
`before.mean`	The treatment mean used for the before sampling. Only really of interest if `method="Lognormal"` as this gives the Shen estimator.

Author(s)

Jon Barry (email Jon.Barry@cefas.co.uk)

References

Shen H, Brown LD and Zhi H (2006) Efficient estimation of log-normal means with application to pharmacokinetic data. Statistics in Medicine, 25, 3023 to 3038.

See Also

`power.trend`, `power.groups`, `size2.samevar`

Examples

```
# Data is richness (number of species) and abundance from grab samples from
# the Dogger Bank, UK. In practice, \code{nsims} would be set to at least 1000.

rich = c(15,16,37,12,15,5,13,16,17,34,23,20,22,30,85,55,13,19,30,41,22,8,43,10,38,24,17,
         23,17,17,24,33,30,18,26,18,12,50,19,21,35)
abun = c(50,91,140,21,25,8,28,37,30,90,56,50,40,83,964,180,21,60,81,138,67,17,250,63,152,
         68,42,69,57,67,74,96,75,44,61,49,62,281,55,50,198)

par(mfrow=c(2,2))
hist(rich)
hist(abun)
hist(sqrt(rich))
hist(log(abun))

ssize = seq(10, 50, 10)
parsc.rich = mean(rich); parst.rich = mean(rich)
parsc.abun = rep(0,2); parst.abun = parsc.abun
parst.abun[1] = mean(log(abun)); parst.abun[2] = sd(log(abun))
parsc.abun = parst.abun
power.rich = rep(0, length(ssize))
power.abun = rep(0, length(ssize))

power.rich = power.BACI(change=35, change.type="M", nt=ssize, nc=ssize,
                parst=parst.rich, parsc=parsc.rich,
                distribution="Poisson", test="P", nsims=50)$power

power.abun = power.BACI(change=35, change.type="M", nt=ssize, nc=ssize,
                parst=parst.abun, parsc=parsc.abun, distribution="Lognormal",
                test="P", nsims=50)$power

par(mfrow=c(1,1))
plot(ssize, power.rich, ylim=c(0,1), ylab="Power", xlab="Sample size", type="l")
lines(ssize, power.abun, lty=2, col=2)
legend("bottomright", legend=c("Richness power", "Abundance power"), lty=c(1,2),
       col=c(1,2))
title("BACI power plots")
```

`power.groups` *Power for comparing mean of two groups*

Description

Calculates the power by simulation for comparing the mean of two groups of independent observations.

Usage

```
power.groups(change, change.type="M", n1, n2, pars1, pars2,
        distribution, test, alternative="two", alpha=0.05, nsims=1000, nreps=999)
```

Arguments

change	Mean of second group minus mean of first group (i.e. mu2-mu1) or percentage change in mu1 to create mu2 (depending on value of change.type).
change.type	Whether the parameter change represents an additive ("A") or percentage ("M") change.
n1	Vector of sample sizes for group 1. Must be of same dimension as n2.
n2	Vector of sample sizes for group 2. Must be of same dimension as n1.
pars1	Parameters for the treatment data. If distribution="Normal", pars1[1] contains the mean for group 1 and pars1[2] contains the standard deviation. If distribution="Poisson", pars1[1] contains the mean for group 1. If distribution="Lognormal", pars1[1] contains the group 1 mean of the natural log data and pars1[2] contains the standard deviation of the log data. If distribution="Negbin", pars1[1] contains the mean and pars1[2] contains the group 1 size parameter.
pars2	pars2[1] is the standard deviation for group 2 if distribution="Normal". If distribution="Lognormal", pars2[1] is the standard deviation of the log data for group 2. For distribution="Negbin", pars2[1] gives the group 2 size.
distribution	The statistical distribution for the two groups. Can be either: "Normal", "Poisson", "Lognormal" or "Negbin".
test	The statistical test used to compare the group means. If test="NP" then the test will be a non-parametric randomisation test, in the spirit of Manly (1997), using the function permute.groups. If test="P", then parametric tests are made to compare the group means. If distribution="Normal", a two sample t-test is carried out. If the standard deviations (defined by pars1[2] and pars2[1] are equal, then the t-test calculates the usual pooled standard deviation. However, if the standard deviations are not equal then the default method used by t.test is adopted. When distribution="Lognormal", natural logs are taken of the simulated data and a t-test used in a similar way as to when distribution="Lognormal". When distribution="Poisson", the difference in deviances between the null model and the model with group membership fitted as factor is compared againt a chi-squared distributiuon on 1 degree of freedom. A similar (but not quite) method is used for when distribution="Negbin". The Generalised Linear Model function glm.nb for the Negative Binomial distribution is used. The p-value for comparing the two groups is taken from the analysis of deviance table after the model with group membership is fitted as a factor. This p-value, however, uses the same value for the size parameter, as estimated from the null model, for group member deviance. This seems to be the correct thing to do as estimating separate size parameters for the two models mucks up the nesting of the models.
alternative	A character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter. As an example, "less" means that the alternative is that the mean of the first group is less than the mean of the second.
alpha	The type 1 error for assessing statistical significance (default is 0.05) in the power simulations.

nsims	Number of repeat simulations to estimate power (default is 1000).
nreps	Number of repeat permutations for randomisation test (default is 999).

Details

The Negative Binomial distribution option allows the user to specify the size parameter for both groups 1 and 2. One possibility is to keep the size the same for both groupss. However, because the mean is different between the groups and because the variance V = mu+mu^2/size, this means that V will be different for the group 1 and group 2 distributions. If you want to keep the variance the same, you can use the function size2.samevar.

Several powers can be calculated per call to this function by specifying more than one values for the sample sizes n1 and n2.

Value

The power is returned. This is the proportion of the nreps simulations that returned a p-value less than the type1 error.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

References

Manly BFJ (1997) Randomization, bootstrap and monte carlo methods in biology: 2nd edition. Chapman and Hall, London, 399 pages.

See Also

permute.groups, glm.nb, size2.samevar

Examples

```
library(MASS)

# In practice, \code{nsims} would be set to at least 1000

power.groups(change=2.5, change.type="A", n1=20, n2=20, pars1=c(10,2),
       pars2=2, test='P', distribution="Normal", nsims=50)

power.groups(change=2.5, change.type="A", n1=seq(5,25,5), n2=seq(5,25,5), pars1=c(10,2),
       pars2=2, test='P', distribution="Normal", nsims=50)

power.groups(change=25, change.type="M", n1=20, n2=20, pars1=10,
       test='P', distribution="Poisson", nsims=50)

power.groups(change=4, change.type="A", n1=20, n2=20, pars1=c(1,2),
       pars2=2, test='P', distribution="Lognormal", nsims=50)


# Keeping size constant
```

```
power.groups(change=100, change.type="M", n1=20, n2=20, pars1=c(5,2),
       pars2=2, test='P', distribution="Negbin", nsims=50)

# Keeping variance constant
s2 = size2.samevar(mu1=5, mu2=10, s1=2)   # 13.333
power.groups(change=100, change.type="M", n1=20, n2=20, pars1=c(5,2),
       pars2=s2, test='P', distribution="Negbin", nsims=50)
```

`power.trend` *Calculates power by simulation to detect a specified trend.*

Description

Calculates power for a specified trend wherethe signal for the trend is specified by xvalues and meanvalues (possibly generated by generate.trend), and the error distribution is specified by distribution. The statistical method to detect the trend is specified by method.The power is the proportion of repeat simulations for which the trend is detected with a p-value less than alpha (two-sided test).

Usage

```
power.trend(xvalues, reps=1,  meanvalues, distribution="Normal", sd=NA,
  nbsize=NA, method="linear regression", alpha=0.05, nsims=1000, nsims.mk=999,
  randeffect=F, randeffect.sd=NA)
```

Arguments

`xvalues`	Vector of, for example, time points at which the trend is evaluated.
`reps`	Vector of number of replicates per time point.
`meanvalues`	Vector of mean values that identify the signal of the trend.
`distribution`	Distribution (must be one of `"Normal"`, `"Poisson"` or `"Negbin"` used to generate random values based on the signal in `meanvalues`.
`sd`	Standard deviation if distribution=`"Normal"`.
`nbsize`	Size parameter if distribution=`"Negbin"`.
`method`	Method used to identify the trend. Can be one of `"linear regression"`, `"mk"`, or `"gam"`. The last of these fits a Generalised Additive Model (Wood, 2006) using function `gam`. It assumes Normal errors.
`alpha`	Type 1 error for detecting trend. Values less than `alpha` cause the null hypothesis of no `trend` to be rejected. Tests are 2-sided.
`nsims`	The number of simulations to be used in calculating the power. Default is 1000.
`nsims.mk`	The number of replicate permutations used in calculating the p-value for the Mann-Kendall test when `method=mk`. Default is 999.
`randeffect`	Not working yet
`randeffect.sd`	Not working yet

Details

The Mann Kendall tests are approriate only for monotonic increasing or decreasing trends, the linear regression method is only approriate for linearly increasing or decreasing trend. The GAM is appropriate for changing trends over time.

Several powers can be calculated on a single call to this function by placing more than one value in `reps`.

Value

The power is returned.

Author(s)

David Maxwell: david.maxwell@cefas.co.uk

References

Fryer RJ & Nicholson MD (1993) Need paper title. ICES Journal of Marine Science, 50, 161-168.

Fryer & Nicholson 1999 Using smoothers for comprehensive assessments of contaminant time series in marine biota. ICES Journal of Marine Science, 56, 779-790.

Wood S.N. (2006) Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC Press.

See Also

`mannkendall.stat`, `addnoise`, `gam`, `generate.trend`

Examples

```
library(mgcv)

# In practice, \code{nsims} would be set to at least 1000

par(mfrow=c(2,2))
lin5 = generate.trend(nyears=10, change=5, type="linear")
plot(lin5$i, lin5$mu)
updown = generate.trend(nyears=15, change=5, type="updown", changeyear=8)
plot(updown$i, updown$mu)

power.trend(xvalues=lin5$i, meanvalues=lin5$mu, distribution="Normal", sd=2,
            method="linear regression", alpha=0.05, nsims=50)
power.trend(xvalues=lin5$i, meanvalues=lin5$mu, distribution="Poisson", method="mk", alpha=0.05,
      nsims=50)
power.trend(xvalues=updown$i, meanvalues=updown$mu, distribution="Normal", sd=2,
            method="gam", alpha=0.05, nsims=50)
```

precision *Sample size for given precision or precision for given sample size*

Description

Precision is measured by the width of a 100(1-alpha) The function generates the sample size needed to achieve this or the precision achieved for a specified sample size.

Usage

```
precision(d, n, pars, method="sample size", alpha=0.05, minint=1, maxint=500)
```

Arguments

d	The Confidence Interval width required (for use with method="sample size"). This can be a vector.
n	Sample size (for use with method="width"). This can be a vector.
pars	Standard deviation of the variable
method	Whether sample size is required ("sample size") or precision ("width").
alpha	Defines the (1-alpha/2) percentage point of the t-dristribution used in the confidence interval.
minint	Lower bound to be used in the search interval for the sample size.
maxint	Upper bound to be used in the search interval for the sample size.

Details

The width of a Confidence Interval for the mean is given by the standard formula d = 2 * sigma * t(1-alpha/2, n-1) / sqrt(n), where sigma is the standard deviation and n is the sample size. t(.) is the relevant quantile of the t distribution function.If sample size is required then we can turn this equation round to get n = [2 * sigma * t(1-alpha/2, n-1)/d]^2. To solve this equation for the sample size n, precision uses the function optimize.

Value

n	Sample sizes.
d	Confidence interval widths.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

Examples

```
precision(d=c(1,1.2,1.5), pars=1, method="sample size", alpha=0.05)

precision(d=c(4), pars=1, method="sample size", alpha=0.05)

precision(n=c(20,25), pars=1, method="width", alpha=0.05)
```

---

size2.samevar *Calculates negative binomial size to preserve constant variance.*

Description

Calculates the Negative Binomial size parameter `s2` such that the variance of the distribution with mean `mu2` and size `s2` is the same as the Negative Binomial distribution with mean `mu1` and size `s1`. This can be useful when computing power for a Negative Binomial distribution in the packages `power.groups` and `power.BACI`.

Usage

```
size2.samevar(mu1, mu2, s1)
```

Arguments

`mu1`	Negative Binomial mean for group 1
`mu2`	Negative Binomial mean for group 2
`s1`	Negative Binomial size for group 1

Value

The size for group 1.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

See Also

`power.groups`, `power.BACI`

Examples

```
mu1=5; mu2=10; s1=3
s2 = size2.samevar(mu1, mu2, s1)
s2
# Check variances are the same
v1 = mu1 + mu1^2 / s1
v2 = mu2 + mu2^2 / s2
v1; v2
```

svariog *Calculates empirical semi-variogram.*

Description

Calculates empirical semi-variogram cloud plus classical, robust and median estimators from bins.

Usage

```
svariog(x, y, z, u)
```

Arguments

x	Location vector 1 (e.g. longitude).
y	Location vector 2 (e.g. latitude).
z	Response vector observed at the locations.
u	(b+1) cut points used to define the b bins for distances. The cut points define the boundaries for each bin. Distances on the boundary of bins go into the lower bin.

Details

Generates the n(n-1)/2 distances between each of the n points together with the semi-variogram cloud of the n(n-1)/2 differences (zi-zj)^2 / 2 between pairs of observations (i,j). This cloud is smoothed by taking one of three sorts of averages within each bin - bin end points are defined by the vector u. These averages are the classical (the bin mean) estimator, a function of the bin median and a robust estimator. Both the median and the robust estimators are based on absolute differences between z pairs. These methods are defined in Cressie (1993).

Value

classical	Classical semi-variogram estimator.
med	Median semi-variogram estimator.
robust	Robust semi-variogram estimator.
freq	Frequencies of distances within each bin.
mid	Mid points of each bin.
zcloud	Unsmoothed semi-variogram cloud.
dcloud	Distances between pairs of points for the variogram cloud.

Author(s)

Jon Barry: Jon.Barry@cefas.co.uk

References

Cressie, NAC (1993) Statistics for Spatial Data, Revised Edition. Wiley, New York.

See Also

variog

Examples

```
# Example based on the number of benthic species found from samples of Hamon Grabs from 50
# locations
lat = c(54.23, 55.14, 55.14, 55.59, 55.49, 55.38, 55.15, 55.14, 55.25, 55.17, 55.16, 54.86,
54.80, 54.95, 54.82, 54.80, 54.80, 54.77, 54.76, 55.48, 55.48, 54.56, 54.55, 54.54, 54.50,
54.63, 54.59, 54.52, 54.40, 54.37, 54.36, 54.16, 55.47, 55.46, 55.12, 55.43, 55.52, 55.62,
55.58, 55.47, 55.35, 55.30, 55.33, 55.32, 55.17, 54.63, 54.95, 54.94, 54.71, 54.36)

long = c(2.730, 1.329, 1.329, 3.225, 1.954, 1.833, 2.090, 2.085, 1.956, 1.643, 1.641, 2.089,
       2.336, 1.489, 1.180, 1.493, 1.493, 1.960, 1.958, 2.559, 2.559, 1.344, 1.343, 1.498,
       1.652, 2.090, 2.331, 2.089, 1.844, 2.335, 2.335, 2.084, 2.903, 2.904, 2.335, 2.335,
       2.338, 2.340, 1.949, 1.469, 1.483, 1.484, 2.901, 2.901, 2.897, 1.040, 1.024, 2.738,
         2.737, 2.551)

nspecies = c(28,16,22,23,17,13,28,18,20,41,21,14,19,41,28,4,32,31,16,9,14,6,35,
          18,9,35,23,5,18,27,27,16,22,16,
          29,11,8,23,28,23,18,16,16,47,31,17,13,23,19,20)

u = c(0,0.1,0.3,0.5,0.7,1,1.5,2.4)

semiv = svariog(long, lat, nspecies, u)

par(mfrow=c(2,2))
plot(semiv$dcloud, semiv$zcloud, xlab='Distance', ylab='Cloud')
plot(semiv$mid, semiv$cla, xlab='Distance', ylab='Classical')
plot(semiv$mid, semiv$med, xlab='Distance', ylab='Median')
plot(semiv$mid, semiv$rob, xlab='Distance', ylab='Robust')
```

Index