Package 'ecopower'

September 6, 2021

Type Package

Title Power Estimates and Equivalence Testing for Multivariate Data

Version 0.1.0

Description Estimates power by simulation for multivariate abundance data to be used for sample size estimates. Multivariate equivalence testing by simulation from a Gaussian copula model. The package also provides functions for parameterising multivariate effect sizes and simulating multivariate abundance data jointly. The discrete Gaussian copula approach is described in Popovic et al. (2018) <doi:10.1016/j.jmva.2017.12.002>.

Depends R (>= 3.5.0)

Imports parallel, mvabund, ecoCopula, stats

Encoding UTF-8

LazyData true

License LGPL (>= 2.1)

RoxygenNote 7.1.1

Suggests testthat

NeedsCompilation no

Author Ben Maslen [aut], Michelle Lim [aut, cre]

Maintainer Michelle Lim <michelle.lim@unsw.edu.au>

Repository CRAN

Date/Publication 2021-09-06 08:10:02 UTC

R topics documented:

`crayweed` *Crayweed dataset*

Description

Dataset of fish abundances recorded at crayweed reference and restored sites.

Usage

```
data(crayweed)
```

Format

An object of class `"list"` containing:

abund A matrix with 27 observations of abundance of 34 fish species.

X A data frame with treatment and time variables.

Details

The matrix abund has the following species abundances:

- Abudefduf.sp.
- Acanthopagrus.australis
- Acanthurus.nigrofuscus
- Achoerodus.viridis
- Aplodactylus.lophodon
- Atypichthys.strigatus
- Cheilodactylus.fuscus
- Chromis.hypsilepis
- Crinodus.lophodon
- Girella.elevata
- Girella.tricuspidata
- Hypoplectrodes.maccullochi
- Needle.fish.unidentified
- Notolabrus.gymnogenis
- Odax.cyanomelas
- Olisthops.cyanomelas
- Ophthalmolepis.lineolatus
- Parma.microlepis
- Parma.unifasciata

- Pempheris.compressa
- Pempheris.multiradiata
- Pictilabrus.laticlavius
- Prionurus.microlepidotus
- Pseudocaranx.dentex
- Pseudojuloides.elongatus
- Pseudolabrus.gymnogenis
- Sardinops.neopilchardus
- Scorpis.lineolatus
- Seriola.lalandi
- Sphyraena.obtusata
- Tetractenos.hamiltoni
- Trachinops.taeniatus
- Trachurus.novaezelandiae
- Upeneichthyes.lineatus

The data frame `X` has the following variables:

- treatment - reference / restored
- time - sample period with seven time points

References

Data attributed to the crayweed restoration project (http://www.operationcrayweed.com/).

Examples

```
data(crayweed)
head(crayweed$abund)
head(crayweed$X)
```

`effect_alt.manyglm` *Specify multivariate effect sizes*

Description

`effect_alt` returns a coefficient matrix to be parsed to `extend`, `powersim` and `equivtest` to specify an effect size of interest.

Usage

```
## S3 method for class 'manyglm'
effect_alt(object, effect_size, increasers, decreasers, term, K = NULL)

effect_alt(object, effect_size, increasers, decreasers, term, K = NULL)
```

Arguments

object	objects of class manyglm, typically the result of a call to manyglm.
effect_size	An effect size of interest, see details for interpretation.
increasers	A vector list of responses which increase relative to the control group/intercept.
decreasers	A vector list of responses which decrease relative to the control group/intercept.
term	Name of predictor of interest in quotes.
K	A vector of length nlevels -1. If NULL, the effect size will increase by its exponent according to the order of factor variables. Alternatively, specify a vector K that corresponds to the exponent of the effect_size for each level of a factor variable. Defaults to NULL, see details.

Details

effect_alt helps users to create interpretable multivariate effect sizes to be parsed into extend, powersim and equivtest, so that researchers can investigate the relationship between effect size, power and sample size in a complicated multivariate abundance setting.

effect_alt creates an effect of size log(effect_size) for a predictor of interest (term), for responses who have been specified to increase (increasers) and -log(effect_size) for responses who have been specified to decrease (decreasers). Responses that have not been specified in the increasers or decreasers vectors are specified to have no effect with a coefficient of 0. The effect has been logged to make the effect size interpretable within the coefficient matrix.

For poisson regression family=poisson() and negative binomial regression family="negative.binomial" the effect size is interpreted for a categorical variable as the multiplicative change in mean abundance in the treatment group relative to the control group, whilst for a continuous variable it is interpreted as the multiplicative change in abundance for a 1 unit increase in the predictor of interest.

For logit regression family=binomial("logit") the effect size is interpreted as an odds ratio. For a categorical variable this is the change in odds of obtaining outcome 1 when being in the treatment group relative to the control group. Whilst for continuous variables, this is interpreted as the change in odds of obtaining outcome 1 with a 1 unit increase in the predictor of interest.

For cloglog regression family=binomial("cloglog") the effect size is interpreted similarly to poisson and negative binomial regression. For a categorical variable it is interpreted as the multiplicative change in the mean of the underlying count in the treatment group relative to the control. Whilst for a continuous variable it is interpreted as the multiplicative change in the mean of the underlying count for a 1 unit increase in the predictor of interest.

For categorical variables, the intercept is also changed to be the group mean intercept by taking the intercept of a model without the categorical predictor of interest. This is done to avoid messy comparisons of null control groups.

For categorical variables with more than two levels, effect size is changed to `effect_size^K[i]` where K defaults to be `c(1,2,...,nlevels -1)`, where `nlevels` are the number of levels of the categorical variable and is specified along the order of the levels. To change this, specify a vector `K` with length of `nlevels -1`. To change the control group, this must be done prior to specifying the `manyglm` object using `relevel` (which can also change the order of the levels).

Note that if the predictor of interest is a categorical variable it must be classed either as a factor or character otherwise results may be misleading.

Value

A coefficient matrix with the specified effect size.

Functions

- `effect_alt`: Specify multivariate effect sizes

See Also

`extend`, `equivtest`, `powersim`

Examples

```
library(mvabund)
data(spider)
spiddat = mvabund(spider$abund)
X = data.frame(spider$x)

# Specify increasers and decreasers
increasers = c("Alopacce", "Arctlute", "Arctperi", "Pardnigr", "Pardpull")
decreasers = c("Alopcune", "Alopfabr", "Zoraspin")

# Obtain an effect matrix of effect_size=3
spid.glm = manyglm(spiddat~soil.dry, family="negative.binomial", data=X)
effect_mat = effect_alt(spid.glm, effect_size=3,
         increasers, decreasers, term="soil.dry")

# Obtain an effect matrix of effect_size=1.5
X$Treatment = rep(c("A","B","C","D"),each=7)
spid.glm = manyglm(spiddat~Treatment, family="negative.binomial", data=X)
effect_mat = effect_alt(spid.glm, effect_size=1.5,
         increasers, decreasers, term="Treatment")

# Change effect size parameterisation
effect_mat = effect_alt(spid.glm, effect_size=1.5,
                        increasers, decreasers, term="Treatment",
                        K=c(3,1,2))
```

`effect_null.manyglm` *Specify null effects for multivariate abundance data*

Description

`effect_null` returns a coefficient matrix to be parsed to `powersim` by default to specify a null effect.

Usage

```
## S3 method for class 'manyglm'
effect_null(object, term)

effect_null(object, term)
```

Arguments

`object`	objects of class `manyglm`, typically the result of a call to `manyglm`.
`term`	Name of predictor of interest in quotes.

Details

`effect_null` produces a coefficient matrix with a null effect that is specified by setting the parameter estimates of a predictor of interest `term` to 0. This function is used by default in `powersim`. Note that intercept values are parameterised as in `effect_alt`.

Value

A coefficient matrix with the null effect.

Functions

- `effect_null`: Specify null effects for multivariate abundance data

See Also

`effect_alt`, `powersim`

Examples

```
library(mvabund)
data(spider)
spiddat = mvabund(spider$abund)
X = data.frame(spider$x)

# Find null effect size for continuous predictor
spid.glm = manyglm(spiddat~soil.dry, family="negative.binomial", data=X)
coeffs0 = effect_null(spid.glm, term="soil.dry")
```

equivtest.cord *Multivariate equivalence testing*

Description

equivtest takes in a copula model fitted to data and a matrix of effect sizes to execute a a multivariate equivalence test.

Usage

```
## S3 method for class 'cord'
equivtest(
  object,
  coeffs,
  term = NULL,
  object0 = NULL,
  stats = NULL,
  test = "LR",
  nsim = 999,
  ncores = detectCores() - 1,
  show.time = TRUE
)

equivtest(
  object,
  coeffs,
  term = NULL,
  object0 = NULL,
  stats = NULL,
  test = "LR",
  nsim = 999,
  ncores = detectCores() - 1,
  show.time = TRUE
)
```

Arguments

object	objects of class cord, typically the result of a call to cord.
coeffs	Coefficient matrix for a manyglm object that characterises the size of effects to be simulated. See effect_alt for help in producing this matrix.
term	Name of predictor of interest in quotes. Defaults to NULL, see details.
object0	object of class cord that specifies the null hypothesis. Defaults to NULL, see details.
stats	Statistics simulated under the null hypothesis. Optional, defaults to NULL. If not NULL, equivtest will not simulate test statistics and use the stats specified.

test	Test statistic for computing p-value. Defaults to "LR".
nsim	Number of simulations for p-value estimate to be based upon. Defaults to 999.
ncores	Number of cores for parallel computing. Defaults to the total number of cores available on the machine minus 1.
show.time	Logical. Displays time elapsed. Defaults to TRUE.

Details

equivtest takes a cord object and a coefficient matrix coeffs which specifies an effect size of interest to perform an equivalence test.

First, marginal parameters of the data are obtained from a manyglm object. Next, a copula model is fitted using cord to estimate the factor analytic covariance structure of the data. The cord function uses two factors by default. The p-value is then obtained by parsing the cord object into extend, nsim times with an effect size specified by coeffs.

The test statistics are simulated under the hypothesis that the effect size equals a certain threshold. The p-value is computed as the proportion of times the simulated test statistics are less than the observed statistic. Equivalence is declared if the estimated effect is less than the threshold.

equivtest can handle any user-defined null hypothesis, so only the fitted null model (object0) or the predictor of interest (term) needs to be specified. If both object0 and term are NULL, equivtest will automatically set the predictor of interest as the last term in the fitted object model or drop the only term in the model to obtain the intercept model.

Simulations are computed in parallel using the "socket" approach, which uses all available cores minus 1 for clustering to improve computation efficiency. Using 1 less than the number of available cores for your machine (detectCores()-1) is recommended to leave one core available for other computer processes.

Value

Equivalence test results, and;

p	p-value;
stat_obs	observed statistic;
stats	simulated statistics.

Functions

- equivtest: Multivariate equivalence testing

See Also

effect_alt

Examples

```
library(ecoCopula)
library(mvabund)
data(spider)
spiddat = mvabund(spider$abund)
X = data.frame(spider$x)

# Specify increasers and decreasers
increasers = c("Alopacce", "Arctlute", "Arctperi", "Pardnigr", "Pardpull")
decreasers = c("Alopcune", "Alopfabr", "Zoraspin")

# Equivalence test for continuous predictor at effect_size=1.5
fit.glm = manyglm(spiddat~bare.sand, family="negative.binomial", data=X)
threshold = effect_alt(fit.glm, effect_size=1.5,
       increasers, decreasers, term="bare.sand")
fit.cord = cord(fit.glm)
equivtest(fit.cord, coeffs=threshold, term="bare.sand", nsim=99, ncores=2)

# Equivalence test for categorical predictor with 4 levels at effect_size=1.5
X$Treatment = rep(c("A","B","C","D"),each=7)
fit_factors.glm = manyglm(spiddat~Treatment, family="negative.binomial", data=X)
threshold = effect_alt(fit_factors.glm, effect_size=1.5,
       increasers, decreasers, term="Treatment")
fit_factors.cord = cord(fit_factors.glm)
equivtest(fit_factors.cord, coeffs=threshold, term="Treatment", nsim=99, ncores=2)

# Specify object0
object0.glm = manyglm(spiddat~1, family="negative.binomial")
object0.cord = cord(object0.glm)
equivtest(fit_factors.cord, coeffs=threshold, object0=object0.cord, nsim=99, ncores=2)
```

`extend.cord` *Simulate or extend multivariate abundance data*

Description

`extend` returns a simulated response matrix or a `manyglm` object with `N` observations and simulated response matrix that utilises the existing correlation structure of the data.

Usage

```
## S3 method for class 'cord'
extend(
  object,
  N = nrow(object$obj$data),
  coeffs = coef(object$obj),
  newdata = NULL,
```

```
  n_replicate = NULL,
  do.fit = FALSE,
  seed = NULL
)

extend(
  object,
  N = nrow(object$obj$data),
  coeffs = coef(object$obj),
  newdata = NULL,
  n_replicate = NULL,
  do.fit = FALSE,
  seed = NULL
)
```

Arguments

object	objects of class cord, typically the result of a call to cord.
N	Number of samples to be extended. Defaults to the number of observations in the original sample.
coeffs	Coefficient matrix for a manyglm object that characterises the size of effects to be simulated. See effect_alt for help in producing this matrix. Defaults to the coefficient matrix from the cord object, coef(object$obj).
newdata	Data frame of same size as the original X covariates from the fitted object, that specifies a different design of interest. Defaults to NULL.
n_replicate	Number of unique replicates of the original data frame. Defaults to NULL, overwrites N if specified.
do.fit	Logical. If TRUE, fits a manyglm object from the simulated data. Defaults to FALSE.
seed	Random number seed, defaults to a random seed number.

Details

extend takes a cord object and returns a new simulated response matrix or an "extended" manyglm object with N observations and the new simulated response matrix. Response abundances are simulated through a Gaussian copula model that utilises a coefficient matrix coeffs, the specified cord model and the joint correlation structure exhibited between the response variables. To help with the specification of coeffs, see effect_alt which simplifies this process.

Response variables are simulated through a copula model by first extracting Gaussian copular scores as Dunn-Smyth residuals (Dunn & Smyth 1996), which are obtained from abundances y_{ij} with marginal distributions F_j which have been specified via the original manyglm model (fit.glm; see examples);

$$z_{ij} = \Phi^{-1} F_j(y_{ij}^-) + u_{ij} f_j(y_{ij})$$

These scores then follow a multivariate Gaussian distribution with zero mean and covariance structure Σ,

$$z_{ij} \sim N_p(0, \Sigma)$$

To avoid estimating a large number $p(p-1)/2$ pairwise correlations within Σ, factor analysis is utilised with two latent factor variables, which can be interpreted as an unobserved environmental covariate.

Thus, in order to simulate new multivariate abundances we simulate new copula scores and back transform them to abundances as $y_{ij} = {F^*}_j^{-1}(\Phi(z_{ij}))$, where the coefficient matrix `coeffs` specifies the effect size within the new marginal distributions ${F^*}_j$.

The data frame is also extended in a manner that preserves the original design structure. This is done by first repeating the design matrix until the number of samples exceeds `N`, then randomly removing rows from the last repeated data frame until the number of samples equals `N`. Alternatively, a balanced design structure can be obtained by specifying the number of replicates.

`newdata` can be utilised if a different data frame is wanted for simulation.

If users are interested in obtaining a `manyglm` model, `do.fit=TRUE` can be used to obtain a `manyglm` object from the simulated responses.

Value

Simulated data or `manyglm` object.

Functions

- `extend`: Simulate or extend multivariate abundance data

References

Dunn, P.K., & Smyth, G.K. (1996). Randomized quantile residuals. Journal of Computational and Graphical Statistics 5, 236-244.

See Also

`effect_alt`

Examples

```
library(ecoCopula)
library(mvabund)
data(spider)
spiddat = mvabund(spider$abund)
X = data.frame(spider$x)

# Specify increasers and decreasers
increasers = c("Alopacce", "Arctlute", "Arctperi", "Pardnigr", "Pardpull")
decreasers = c("Alopcune", "Alopfabr", "Zoraspin")

# Simulate data
fit.glm = manyglm(spiddat~1, family="negative.binomial")
```

```
fit.cord = cord(fit.glm)
simData = extend(fit.cord)

# Simulate data with N=20
fit.glm = manyglm(spiddat~soil.dry, family="negative.binomial", data=X)
fit.cord = cord(fit.glm)
simData = extend(fit.cord, N=20)

# Obtain a manyglm fit from simulated data with N=10 and effect_size=1.5
X$Treatment = rep(c("A","B","C","D"),each=7)
fit_factors.glm = manyglm(spiddat~Treatment, family="negative.binomial", data=X)
effect_mat = effect_alt(fit_factors.glm, effect_size=1.5,
     increasers, decreasers, term="Treatment")
fit_factors.cord = cord(fit_factors.glm)
newFit.glm = extend(fit_factors.cord, N=10,
     coeffs=effect_mat, do.fit=TRUE)

# Change sampling design
X_new = X
X_new$Treatment[6:7] = c("B","B")
simData = extend(fit_factors.cord, N=NULL,
   coeffs=effect_mat, newdata=X_new, n_replicate=5)
```

`powersim` *Provide power estimates for multivariate abundance models*

Description

`powersim` returns a power estimate for a `cord` object for a given sample size `N` and effect size of interest.

Usage

```
powersim(
  object,
  coeffs,
  term,
  N = nrow(object$obj$data),
  coeffs0 = effect_null(object$obj, term),
  nsim = 999,
  test = "score",
  alpha = 0.05,
  newdata = NULL,
  n_replicate = NULL,
  ncores = detectCores() - 1,
  show.time = TRUE
)

## S3 method for class 'cord'
```

```
powersim(
  object,
  coeffs,
  term,
  N = nrow(object$obj$data),
  coeffs0 = effect_null(object$obj, term),
  nsim = 999,
  test = "score",
  alpha = 0.05,
  newdata = NULL,
  n_replicate = NULL,
  ncores = detectCores() - 1,
  show.time = TRUE
)
```

Arguments

object	objects of class cord, typically the result of a call to cord.
coeffs	Coefficient matrix for a manyglm object that characterises the size of effects to be simulated. See effect_alt for help in producing this matrix.
term	Name of predictor of interest in quotes.
N	Number of samples for power estimate. Defaults to the number of observations in the original sample.
coeffs0	Coefficient matrix under the null hypothesis. Defaults to being specified by effect_null.
nsim	Number of simulations for power estimate to be based upon. Defaults to 999.
test	Test statistic for power estimate to based upon. Defaults to "score", however "wald" is also allowed.
alpha	Type I error rate for power estimate, defaults to 0.05.
newdata	Data frame of the same size as the original data frame from the cord object (objectobjdata), that specifies a different design of interest.
n_replicate	Number of unique replicates of the original data frame. Defaults to NULL, overwrites N if specified.
ncores	Number of cores for parallel computing. Defaults to the total number of cores available on the machine minus 1.
show.time	Logical. Displays time elapsed. Defaults to TRUE.

Details

powersim takes a cord object, sample size N and coefficient matrix coeffs which specifies an effect size of interest and returns a power estimate.

The power estimate is obtained by first parsing the cord object into extend, nsim times with an effect size specified by coeffs. Next, the cord object is parsed into extend an additional nsim times with a null effect, which is defined by default by effect_null. This effectively simulates nsim manyglm models under both the null and alternative hypothesis.

For each simulated manyglm object a test statistic test is obtained. A critical test statistic is then obtained as the upper 1 - alpha quantile of simulated test statistics under the null hypothesis. Power is then estimated as the proportion of times the test statistics simulated under the alternative hypothesis exceed the critical test statistic under the null.

To improve computation time, simulations are computed in parallel using the "socket" approach, which by default uses all available cores minus 1 for clustering. Using 1 less than the number of available cores for your machine (detectCores()-1) is recommended to better avoid error relating to clustering or nodes.

Value

Power estimate result, and;

power power.

Functions

- powersim: Provide power estimates for multivariate abundance models

See Also

effect_alt, effect_null, extend

Examples

```
library(ecoCopula)
library(mvabund)
data(spider)
spiddat = mvabund(spider$abund)
X = data.frame(spider$x)

# Specify increasers and decreasers
increasers = c("Alopacce", "Arctlute", "Arctperi", "Pardnigr", "Pardpull")
decreasers = c("Alopcune", "Alopfabr", "Zoraspin")

# Find power for continuous predictor at effect_size=1.5
fit.glm = manyglm(spiddat~bare.sand, family="negative.binomial", data=X)
effect_mat = effect_alt(fit.glm, effect_size=1.5,
       increasers, decreasers, term="bare.sand")
fit.cord = cord(fit.glm)
powersim(fit.cord, coeffs=effect_mat, term="bare.sand", nsim=99, ncores=2)

# Find power for categorical predictor with 4 levels at effect_size=1.5
X$Treatment = rep(c("A","B","C","D"),each=7)
fit_factors.glm = manyglm(spiddat~Treatment, family="negative.binomial", data=X)
effect_mat = effect_alt(fit_factors.glm, effect_size=1.5,
       increasers, decreasers, term="Treatment")
fit_factors.cord = cord(fit_factors.glm)
powersim(fit_factors.cord, coeffs=effect_mat, term="Treatment", nsim=99, ncores=2)

# Change effect size parameterisation
```

```
effect_mat = effect_alt(fit_factors.glm, effect_size=1.5,
                        increasers, decreasers, term="Treatment",
                        K=c(3,1,2))
powersim(fit_factors.cord, coeffs=effect_mat, term="Treatment", nsim=99, ncores=2)
```

Index