Package 'mvabund'

January 25, 2017

Title Statistical Methods for Analysing Multivariate Abundance Data

Version 3.12

Date 2016-10-06

Author Yi Wang, Ulrike Naumann, Stephen Wright, Dirk Eddelbuettel and David Warton

Maintainer David Warton <David.Warton@unsw.edu.au>

Description A set of tools for displaying, modeling and analysing multivariate abundance data in community ecology. See 'mvabund-package.Rd' for details of overall package organization. The package is implemented with the Gnu Scientific Library (http://www.gnu.org/software/gsl/) and Rcpp (http://dirk.eddelbuettel.com/code/rcpp.html) R / C++ classes.

Depends R (>= 3.0.0)

Imports Rcpp, MASS, methods, stats, tweedie, statmod, parallel

LinkingTo Rcpp, RcppGSL

License LGPL (>= 2.1)

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-01-25 09:03:06

R topics documented:

mvabund-package 2
anova.manyany 5
anova.manyglm 8
anova.manylm 13
anova.traitglm 17
antTraits 19
best.r.sq 20
boxplot.mvabund 22
cv.glm1path 25
deviance.manylm 27
extend.x.formula 28

formulaUnimva . . . 29
glm1 . . . 30
glm1path . . . 32
logLik.manylm . . . 35
manyany . . . 36
manyglm . . . 39
manylm . . . 44
manylm.fit . . . 47
meanvar.plot . . . 48
mvabund . . . 50
mvformula . . . 52
plot.manyany . . . 53
plot.manylm . . . 54
plot.mvabund . . . 57
plotMvaFactor . . . 62
predict.manyglm . . . 63
predict.manylm . . . 65
predict.traitglm . . . 67
residuals.manyglm . . . 68
ridgeParamEst . . . 70
shiftpoints . . . 71
solberg . . . 72
spider . . . 73
summary.manyglm . . . 75
summary.manylm . . . 80
Tasmania . . . 85
tikus . . . 87
traitglm . . . 88
unabund . . . 91

Index **93**

`mvabund-package` *Statistical methods for analysing multivariate abundance data*

Description

This package provides tools for a model-based approach to the analysis of multivariate abundance data in ecology (Warton 2011), where 'abundance' should be interpreted loosely - as well as counts you could have presence/absence, ordinal or biomass (via `manyany`), etc.

There are graphical methods for exploring the properties of data and the community-environment association, flexible regression methods for estimating and making robust inferences about the community-environment association, 'fourth corner models' to explain environmental response as a function of traits, and diagnostic plots to check the appropriateness of a fitted model (Wang et. al 2012).

There is an emphasis on design-based inferences about these models, e.g. bootstrapping rows of residuals via `anova` calls, or cross-validation across rows, to make multivariate inferences that are

robust to failure of assumptions about correlation. Another emphasis is on presenting diagnostic tools to check assumptions, especially via residual plotting.

Details

The key functions available in this package are the following.

For graphical display of the data:

`plot.mvabund` draw a range of plots for Multivariate Abundance Data

`boxplot.mvabund` draw a range of plots of Model Formulae for Multivariate Abundance Data

`meanvar.plot` draw mean-variance plots for Multivariate Abundance Data

For estimating and displaying Linear Models:

`manylm` Fitting Linear Models for Multivariate Abundance Data

`summary.manylm` summarizie Multivariate Linear Model Fits for Abundance Data

`anova.manylm` obtain ANOVA for Multivariate Linear Model Fits for Abundance Data

`plot.manylm` plot diagnostics for a `manylm` Object

For estimating and displaying Generalized Linear Models:

`manyglm` fit Generalized Linear Models for Multivariate Abundance Data

`summary.manyglm` summarize Multivariate Generalized Linear Model Fits for Abundance Data

`anova.manyglm` obtain Analysis of Deviance for Multivariate Generalized Linear Model Fits for Abundance Data

`plot.manyglm` plot diagnostics for a `manyglm` Object

Other generic functions like `residuals`, `predict`, `AIC` can be applied to `manyglm` objects.

For estimating and displaying 'fourth corner models' with species traits as well as environmental predictors:

`traitglm` predict abundance using a GLM as a function of traits as well as environmental variables

`anova.traitglm` obtain Analysis of Deviance for a fourth corner model of abundance

Other generic functions like `plot`, `residuals`, `predict`, `AIC` can be applied to `traitglm` objects. Note `traitglm` can work slowly, as it fits a single big model to vectorised data (then wants to resample it when you call `anova.traitglm`).

For fitting more flexible models:

`manyany` simultaneously fit univariate models to each response variable from 'any' input function

`anova.manyany` simultaneously test for a community-level effect, comparing two or more `manyany` objects

`glm1path` fit a path of Generalised Linear Models with L1 ('LASSO') penalties

`cv.glm1path` choose the value of the L1 penalty in a `glm1path` fit by cross-validation

Other generic functions like `residuals`, `predict`, `AIC` can be applied to `manyany` and `glm1path` objects. These functions also can be on the slow side, especially if all rare species are included.

For providing a data structure:

`mvabund` create a mvabund object

`mvformula` create Model Formulae for Multivariate Abundance Data

Example datasets:

`Tasmania` meiobenthic community data from Tasmania. Used to demonstrate test for interaction.

`solberg` solberg species counts with a 3-level treatment factor.

`spider` hunting spiders counts from different sites.

`tikus` solberg nematode counts from Tikus island.

`antTraits` ant counts from Eucalypt forests, with trait measurements.

For more details, see the documentation for any of the individual functions listed above.

Author(s)

David Warton <David.Warton@unsw.edu.au>, Yi Wang and Ulrike Naumann.

References

Brown AM, Warton DI, Andrew NR, Binns M, Cassis G and Gibb H (2014) The fourth corner solution - using species traits to better understand how species traits interact with their environment, *Methods in Ecology and Evolution* 5, 344-352.

Warton D.I. (2008a). Raw data graphing: an informative but under-utilized tool for the analysis of multivariate abundances. *Austral Ecology* 33, 290-300.

Warton D.I. (2008b). Penalized normal likelihood and ridge regularization of correlation and covariance matrices. *Journal of the American Statistical Association* 103, 340-349.

Warton D.I. (2011). Regularized sandwich estimators for analysis of high dimensional data using generalized estimating equations. *Biometrics*, 67, 116-123.

Warton DI, Shipley B & Hastie T (2015) CATS regression - a model-based approach to studying trait-based community assembly, *Methods in Ecology and Evolution* 6, 389-398.

Warton D. I., Wright S., and Wang, Y. (2012). Distance-based multivariate analyses confound location and dispersion effects. *Methods in Ecology and Evolution*, 3, 89-101.

Wang Y., Neuman U., Wright S. and Warton D. I. (2012). mvabund: an R package for model-based analysis of multivariate abundance data. *Methods in Ecology and Evolution*, 3, 471-473.

See Also

`plot.mvabund`, `meanvar.plot`, `manyany`, `manylm`, `manyglm`, `traitglm`, `summary.manylm`, `anova.manyany`, `anova.manylm`, `anova.traitglm`, `anova.manyglm`, `plot.manylm`

Examples

```
require(graphics)

## Load the spider dataset:
data(spider)
```

```
## Create the mvabund object spiddat:
spiddat <- mvabund(spider$abund)
X <- spider$x

## Draw a plot of the spider data:
plot(spiddat, col="gray1", n.vars=8, transformation="sqrt",
xlab=c("Hunting Spider"), ylab="Spider Species", scale.lab="s",
t.lab="t", shift=TRUE, fg= "lightblue", col.main="red", main="Spiders")


## A mean-variance plot, data organised by year,
## for 1981 and 1983 only, as in Figure 7a of Warton (2008a):
data(tikus)
tikusdat <- mvabund(tikus$abund)
year <- tikus$x[,1]
is81or83 <- year==81 | year==83
meanvar.plot(tikusdat~year,legend=TRUE, subset=is81or83, col=c(1,10))

## Create a formula for multivariate abundance data:
foo <- mvformula( spiddat~X )

## Create a List of Univariate Formulas:
fooUni <- formulaUnimva(spiddat~X)
fooUniInt <- formulaUnimva(spiddat~X, intercept=TRUE)

## Find the three variables that best explain the response:
best.r.sq( foo, n.xvars= 3)

## Fit a multivariate linear model:
foo <- mvformula( spiddat~X )
lm.spider <- manylm(foo)

## Plot Diagnostics for a multivariate linear model:
plot(lm.spider,which=1:2,col.main="red",cex=3,overlay=FALSE)

## Obtain a summary of test statistics using residual resampling:
summary(lm.spider, nBoot=500)

## Calculate a ANOVA Table:
anova(lm.spider, nBoot=500)
```

---

anova.manyany *Analysis of Deviance for Many Univariate Models Fitted to Multivariate Abundance Data*

Description

Compute an analysis of deviance table for many univariate model fits. Slowly!

Usage

```
## S3 method for class 'manyany'
anova(object, ..., nBoot=99, p.uni="none", block=object1$block, nCores=1,
       bootID=NULL, replace=TRUE)

## S3 method for class 'anova.manyany'
print(x, ...)
```

Arguments

object	of class manyany under the null hypothesis, typically the result of a call to manyany.
...	other generic anova methods. NEEDS TO INCLUDE A SECOND manyany object for the alternative hypothesis to be tested.
nBoot	the number of Bootstrap iterations, default is nBoot=99.
p.uni	whether to calculate univariate test statistics and their P-values. "none" = No univariate P-values (default) "unadjusted" = A test statistic and (ordinary unadjusted) P-value are reported for each response variable. If the manyany object is compositional (composition=TRUE), this option is unavailable as yet.
block	a factor specifying the sampling level to be resampled. Default is resampling rows (if composition=TRUE in the manyany command, this means resampling rows of data as originally sent to manyany).
nCores	Number of cores to use for computations (for parallel computing).
bootID	A user-entered matrix of indices for which observations to use in which resample. Bootstrap resamples in rows, observations in columns. When specified, overwrites nBoot and block. Default is NULL.
replace	whether to sample with or without replacement, as in the sample function. = FALSE for PIT-permutation, = TRUE for PIT-trap.
x	anova.manyany object to be printed.

Details

The anova.manyany function returns a table summarising the statistical significance of a fitted manyany model under the alternative hypothesis (object2) as compared to a fit under the null hypothesis (object). Typically the alternative model is nested in the null although it doesn't need to be (but consider seriously if what you are doing makes sense if they are not nested).

This function is quite computationally intensive, and a little fussy - it is an early version we hope to improve on. Feedback welcome!

This function behaves a lot like anova.manyglm, the most conspicuous differences being in flexibility and computation time. Since this function is based on manyany, it offers much greater flexibility in terms of types of models that can be fitted (most fixed effects model with predict and family arguments could be accommodated). For information on the different types of data that can be modelled using manyany, see manyany.

However this flexibility comes at considerable cost in terms of computation time, and the default nBoot has been set to 99 to reflect this (although rerunning at 999 is recommended). Other more cosmetic differences from anova.manyglm are that two and only two models can be supplied as input here; adjusted univariate P-values are not yet implemented; and the range of test statistics and resampling algorithms is more limited. All test statistics constructed here are sum-of-likelihood ratio statistics as in Warton et al (2012), and the resampling method used here is the PIT-trap (short for 'probability integral transform residual bootstrap').

To check model assumptions, use plot.manyany.

The block argument allows for block resampling, such that valid inferences can be made across independent blocks of correlated sets of observations. For example, if data have multiple rows of records for each site, e.g. multi-species data with entries for different species on different rows, you can use your site ID variable as the block argument to resample sites, for valid cross-site inferences despite within-site species correlation. Well, valid assuming sites are independent. You could do similarly for a repeated measures design to make inferences robust to temporal autocorrelation. Note that block needs to be balanced, e.g. equal number of species entries for each site (i.e. include rows for zero abundances too).

The anova.manyany function is designed specifically for high-dimensional data (that, is when the number of variables p is not small compared to the number of observations N). In such instances a correlation matrix is computationally intensive to estimate and is numerically unstable, so by default the test statistic is calculated assuming independence of variables. Note however that the resampling scheme used ensures that the P-values are approximately correct even when the independence assumption is not satisfied.

Rather than stopping after testing for multivariate effects, it is often of interest to find out which response variables express significant effects. Univariate statistics are required to answer this question, and these are reported if requested. Setting p.uni="unadjusted" returns resampling-based univariate P-values for all effects as well as the multivariate P-values, if composition=FALSE. There are currently no univariate P-value options when composition=TRUE (it's not entirely clear how such P-values should be obtained) and if univariate P's are of interest why not rerun the model with composition=FALSE.

Value

stat	the observed value of the test statistic.
p	the P-value as estimated from nBoot resamples.
stat.i	the values of the test statistic in each of the nBoot resamples.
p.i	the P-value in each of the nBoot resamples.
p.uni	the p.uni argument supplied.

If p.uni="unadjusted" the output list also contains

uni.test	a table showing the test statistics of the univariate tests.
uni.p	a table showing the p-values of the univariate tests.
statj.i	a matrix of values of the univariate test statistics in each of the nBoot resamples.

Warning

The comparison between two or more models by `anova.manyglm` will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values and R's default of `na.action = na.omit` is used.

Author(s)

David Warton <David.Warton@unsw.edu.au>.

References

Warton D. I., Wright S., and Wang, Y. (2012). Distance-based multivariate analyses confound location and dispersion effects. *Methods in Ecology and Evolution*, 3(1), 89-101.

See Also

`manyany`, `anova.manyglm`.

Examples

```
## Try fitting Tikus Islands data with Tweedie models with power parameter 1.5,
## to test for compositional effect:
data(tikus)
coral <- as.matrix(tikus$abund[1:20,])
sumSpp = apply(coral>0,2,sum)

coral <- coral[,sumSpp>6] ## cutting to just species with seven(!) or more presences to cut
## computation time.  Maybe rerun with less (e.g. 4 or more presences) if curious and patient.
coralX <- tikus$x[1:20,]

require(tweedie)
require(statmod)

ftTimeRep <- manyany("glm", coral, coral ~ time+rep, data=coralX,
family=tweedie(var.power=1.5, link.power=0), var.power=1.5, composition=TRUE)

ftRep <- manyany("glm",coral, coral ~ rep, data=coralX,
family=tweedie(var.power=1.5, link.power=0), var.power=1.5, composition=TRUE)
anova(ftRep,ftTimeRep,nBoot=9) #this takes a few seconds to run even for just 9 resamples
## This should be rerun for nBoot=999, which would take maybe five minutes...
```

`anova.manyglm` *Analysis of Deviance for Multivariate Generalized Linear Model Fits for Abundance Data*

Description

Compute an analysis of deviance table for one or more multivariate generalized linear model fits.

Usage

```
## S3 method for class 'manyglm'
anova(object, ..., resamp="pit.trap", test="LR", p.uni="none",
    nBoot=999, cor.type=object$cor.type, block=NULL, show.time="total",
    show.warning=FALSE, rep.seed=FALSE, bootID=NULL)
## S3 method for class 'anova.manyglm'
print(x, ...)
```

Arguments

object	objects of class manyglm, typically the result of a call to manyglm.
...	for the anova.manyglm method, these are optional further objects of class manyglm, which are usually a result of a call to manyglm for the print.anova.manyglm method these are optional further arguments passed to or from other methods. See print.summary.glm for more details.
resamp	the method of resampling used. Can be one of "perm.resid", "montecarlo" or "pit.trap" (default). See Details.
test	the test to be used. If cor.type="I", this can be one of "wald" for a Wald-Test or "score" for a Score-Test or "LR" for a Likelihood-Ratio-Test, otherwise only "wald" and "score" is allowed. The default value is "LR".
p.uni	whether to calculate univariate test statistics and their P-values, and if so, what type. This can be one of the following options. "none" = No univariate P-values (default) "unadjusted" = A test statistic and (ordinary unadjusted) P-value are reported for each response variable. "adjusted" = Univariate P-values are adjusted for multiple testing, using a step-down resampling procedure.
nBoot	the number of Bootstrap iterations, default is nBoot=999.
cor.type	structure imposed on the estimated correlation matrix under the fitted model. Can be "I"(default), "shrink", or "R". See Details.
block	a factor specifying the sampling level to be resampled. Default is resampling rows.
show.time	Whether to display timing information for the resampling procedure: "none" shows none, "all" shows all timing information and "total" shows only the overall time taken for the tests.
show.warning	logical. Whether to display warning messages in the operation procedure.
rep.seed	logical. Whether to fix random seed in resampling data. Useful for simulation or diagnostic purposes.
bootID	an integer matrix where each row specifies bootstrap id's in each resampling run. When bootID is supplied, nBoot is set to the number of rows in bootID. Default is NULL.
x	an object of class "anova.manyglm", usually, a result of a call to anova.manyglm.

Details

The anova.manyglm function returns a table summarising the statistical significance of a fitted manyglm model (Warton 2011), or of the differences between several nested models. If one model is specified, sequential test statistics (and P values) are returned for that fit. If more than one object is specified, the table contains test statistics (and P values) comparing their fits, provided that the models are fitted to the same dataset.

The test statistics are determined by the argument test, and the P-values are calculated by resampling rows of the data using a method determined by the argument resamp. resamp. Two of the three available resampling methods (residual permutation and parametric bootstrap) are described in more detail in Davison and Hinkley (1997, chapter 6), whereas the default (the "PIT-trap") is a new method (in review) which bootstraps probability integral transform residuals, and which we have found to give the most reliable Type I error rates. All methods involve resampling under the resampling under the null hypothesis. These methods ensure approximately valid inference even when the mean-variance relationship or the correlation between variables has been misspecified. Standardized Pearson residuals (see manyglm are currently used in residual permutation, and where necessary, resampled response values are truncated so that they fall in the required range (e.g. counts cannot be negative). However, this can introduce bias, especially for family=binomial, so we advise extreme caution using perm.resid for presence/absence data. If resamp="none", p-values cannot be calculated, however the test statistics are returned.

If you do not have a specific hypothesis of primary interest that you want to test, and are instead interested in which model terms are statistically significant, then the summary.manyglm function is more appropriate. Whereas summary.manyglm tests the significance of each explanatory variable, anova.manyglm, given one manyglm object tests each term of the formula, e.g. if the formula is 'y~a+b' then a and b, that can be vectors or matrices, are tested for significance.

For information on the different types of data that can be modelled using manyglm, see manyglm. To check model assumptions, use plot.manyglm.

Multivariate test statistics are constructed using one of three methods: a log-likelihood ratio statistic test="LR", for example as in Warton et. al. (2012) or a Wald statistic test="wald" or a Score statistic test="score". "LR" has good properties, but is only available when cor.type="I".

The default Wald test statistic makes use of a generalised estimating equations (GEE) approach, estimating the covariance matrix of parameter estimates using a sandwich-type estimator that assumes the mean-variance relationship in the data is correctly specified and that there is an unknown but constant correlation across all observations. Such assumptions allow the test statistic to account for correlation between variables but to do so in a more efficient way than traditional GEE sandwich estimators (Warton 2011). The common correlation matrix is estimated from standardized Pearson residuals, and the method specified by cor.type is used to adjust for high dimensionality.

The Wald statistic has problems for count data and presence-absence data when there are estimated means at zero (which usually means very large parameter estimates, check for this using coef). In such instances Wald statistics should not be used, Score or LR should do the job.

The anova.manyglm function is designed specifically for high-dimensional data (that, is when the number of variables p is not small compared to the number of observations N). In such instances a correlation matrix is computationally intensive to estimate and is numerically unstable, so by default the test statistic is calculated assuming independence of variables (cor.type="I"). Note however that the resampling scheme used ensures that the P-values are approximately correct even when the independence assumption is not satisfied. However if it is computationally feasible for your dataset, it is recommended that you use cor.type="shrink" to account for correlation between variables,

or cor.type="R" when p is small. The cor.type="R" option uses the unstructured correlation matrix (only possible when N>p), such that the standard classical multivariate test statistics are obtained. Note however that such statistics are typically numerically unstable and have low power when p is not small compared to N.

The cor.type="shrink" option applies ridge regularisation (Warton 2008), shrinking the sample correlation matrix towards the identity, which improves its stability when p is not small compared to N. This provides a compromise between "R" and "I", allowing us to account for correlation between variables, while using a numerically stable test statistic that has good properties.

The shrinkage parameter is an attribute of a manyglm object. For a Wald test, the sample correlation matrix of the alternative model is used to calculate the test statistics. So shrink.param of the alternative model is used. For a score test, the sample correlation matrix of the null model is used to calculate the test statistics. So shrink.param of the null model is used instead. If cor.type=="shrink" and shrink.param is NULL, then the shrinkage parameter will be estimated by cross-validation using the multivariate normal likelihood function (see ridgeParamEst and (Warton 2008)) for the corresponding model in the anova test.

Rather than stopping after testing for multivariate effects, it is often of interest to find out which response variables express significant effects. Univariate statistics are required to answer this question, and these are reported if requested. Setting p.uni="unadjusted" returns resampling-based univariate P-values for all effects as well as the multivariate P-values, whereas p.uni="adjusted" returns adjusted P-values (that have been adjusted for multiple testing), calculated using a step-down resampling algorithm as in Westfall & Young (1993, Algorithm 2.8). This method provides strong control of family-wise error rates, and makes use of resampling (using the method controlled by resamp) to ensure inferences take into account correlation between variables.

Value

family	the family component from object.
p.uni	the p.uni argument supplied.
test	the test argument supplied.
cor.type	the cor.type argument supplied.
resamp	the resamp argument supplied.
nBoot	the nBoot argument supplied.
shrink.parameter	a list of shrink parameters from all manyglm objects in the anova test.
n.bootsdone	the number of bootstrapping iterations that were done, i.e. had no error.
table	the table with Residual Degrees of Freedom, Degrees of Freedom, the Test Statistics and the P values.
block	any block argument specified as an inpout argument.

If p.uni="adjusted" or "unadjusted" the output list also contains

uni.test	a table showing the test statistics of the univariate tests.
uni.p	a table showing the p-values of the univariate tests.

Warning

The comparison between two or more models by `anova.manyglm` will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values and R's default of `na.action = na.omit` is used.

Author(s)

Yi Wang, Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

References

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and their Application.* Cambridge University Press, Cambridge.

Warton D.I. (2011). Regularized sandwich estimators for analysis of high dimensional data using generalized estimating equations. *Biometrics*, 67(1), 116-123.

Warton D.I. (2008). Penalized normal likelihood and ridge regularization of correlation and covariance matrices. *Journal of the American Statistical Association* 103, 340-349.

Warton D. I., Wright S., and Wang, Y. (2012). Distance-based multivariate analyses confound location and dispersion effects. *Methods in Ecology and Evolution*, 3(1), 89-101.

Westfall, P. H. and Young, S. S. (1993) *Resampling-based multiple testing.* John Wiley & Sons, New York.

Wu, C. F. J. (1986) Jackknife, Bootstrap and Other Resampling Methods in Regression Analysis. *The Annals of Statistics* 14:4, 1261-1295.

See Also

`manyglm`, `summary.manyglm`.

Examples

```
## Load the Tasmania data set
data(Tasmania)

## Visualise the effect of treatment on copepod abundance
tasm.cop <- mvabund(Tasmania$copepods)
treatment <- Tasmania$treatment
block <- Tasmania$block
#plot(tasm.cop ~ treatment, col=as.numeric(block))

## Fitting predictive models using a negative binomial model for counts:
tasm.cop.nb <- manyglm(tasm.cop ~ block*treatment, family="negative.binomial")

## Testing hypotheses about the treatment effect and treatment-by-block interactions,
## using a Wald statistic and 199 resamples (better to ramp up to 999 for a paper):
anova(tasm.cop.nb, nBoot=199, test="wald")
```

anova.manylm *ANOVA for Linear Model Fits for Multivariate Abundance Data*

Description

anova method for class "manylm" - computes an analysis of variance table for one or more linear model fits to high-dimensional data, such as multivariate abundance data in ecology.

Usage

```
## S3 method for class 'manylm'
anova(object, ..., resamp="perm.resid", test="F", p.uni="none",
        nBoot=999, cor.type=object$cor.type, shrink.param=object$shrink.param,
studentize=TRUE, calc.rss = FALSE, tol=1.0e-10, rep.seed=FALSE, bootID=NULL )
## S3 method for class 'anova.manylm'
print(
    x, digits = max(getOption("digits") - 3, 3),
    signif.stars = getOption("show.signif.stars"),
    dig.tst = max(1, min(5, digits - 1)),
    eps.Pvalue = .Machine$double.eps, ...)
```

Arguments

object	object of class manylm, usually, a result of a call to manylm.
...	for the anova.manylm method, these are optional further objects of class manylm, which are usually a result of a call to manylm. for the print.anova.manylm method these are optional further arguments passed to or from other methods.
nBoot	the number of iterations in resampling. Default is 999 for P-values as fractions out of 1000.
resamp	the method of resampling used. Can be one of "case" (not yet available), "residual" (default), "perm.resid", "score" or "none". See Details.
test	the test to be used. Possible values are: "LR" = likelihood ratio statistic, "F" = Lawley-Hotelling trace statistic or NULL for no test.
cor.type	structure imposed on the estimated correlation matrix under the fitted model. Can be "I"(default), "shrink", or "R". See Details.
shrink.param	shrinkage parameter to be used if cor.type="shrink". If not supplied, but needed, it will be estimated by estimated from the data by Cross Validation using the normal likelihood as in Warton (2008).
p.uni	whether to calculate univariate test statistics and their P-values, and if so, what type. "none" = no univariate P-values (default) "unadjusted" = A test statistic and (ordinary unadjusted) P-value is reported for each response variable. "adjusted" = Univariate P-values are adjusted for multiple testing, using a step-down resampling procedure.

studentize	logical. Whether studentized residuals should be used to simulate the data in the resampling steps. This option is not used in case resampling.
calc.rss	logical. Whether the Residual Sum of Squares should be calculated.
tol	the sensitivity in calculations near 0.
rep.seed	logical. Whether to fix random seed in resampling data. Useful for simulation or diagnostic purposes.
bootID	an integer matrix where each row specifies bootstrap id's in each resampling run. When bootID is supplied, nBoot is set to the number of rows in bootID. Default is NULL.
x	an object of class "anova.manylm", usually, a result of a call to anova.manylm.
digits	the number of significant digits to use when printing.
signif.stars	logical. If TRUE, 'significance stars' are printed for each coefficient.
dig.tst	the number of digits to round the estimates of the model parameters.
eps.Pvalue	a numerical tolerance for the formatting of p values.

Details

The anova.manylm function returns a table summarising the statistical significance of a fitted manylm model, or of the differences between several nested models fitted to the same dataset. If one model is specified, sequential test statistics (and P values) are returned for that fit. If more than one object is specified, the table contains test statistics (and P values) comparing their fits.

The test statistics are determined by the argument test, and the P-values are calculated by resampling rows of the data using a method determined by the argument resampling. The four possible resampling methods are residual-permutation (Anderson and Robinson (2001)), score resampling (Wu (1986)), case and residual resampling (Davison and Hinkley (1997, chapter 6)), and involve resampling under the null hypothesis (except for case resampling). These methods ensure approximately valid inference even when the correlation between variables has been misspecified, and for case and score resampling, even when the equal variance assumption of linear models is invalid. By default, studentised residuals (r_i/sqrt(1-h_ii)) are used in residual and score resampling, although raw residuals could be used via the argument studentize=FALSE. If resamp="none", p-values cannot be calculated, however the test statistics are returned.

If you do not have a specific hypothesis of primary interest that you want to test, and are instead interested in which model terms are statistically significant, then the summary.manylm function is more appropriate.

More than one object should only be specified when the models are nested. In this case the ANOVA table has a column for the residual degrees of freedom and a column for change in degrees of freedom. It is conventional to list the models from smallest to largest, but this is up to the user.

To check model assumptions, use plot.manylm.

The anova.manylm function is designed specifically for high-dimensional data (that, is when the number of variables p is not small compared to the number of observations N). In such instances a correlation matrix is computationally intensive to estimate and is numerically unstable, so by default the test statistic is calculated assuming independence of variables (cor.type="I"). Note however that the resampling scheme used ensures that the P-values are approximately correct even when the independence assumption is not satisfied. However if it is computationally feasible for your dataset,

it is recommended that you use `cor.type="shrink"` to account for correlation between variables, or `cor.type="R"` when p is small. The `cor.type="R"` option uses the unstructured correlation matrix (only possible when N>p), such that the standard classical multivariate test statistics are obtained. Note however that such statistics are typically numerically unstable and have low power when p is not small compared to N. The `cor.type="shrink"` option applies ridge regularisation (Warton 2008), shrinking the sample correlation matrix towards the identity, which improves its stability when p is not small compared to N. This provides a compromise between `"R"` and `"I"`, allowing us to account for correlation between variables, while using a numerically stable test statistic that has good properties. The shrinkage parameter by default is estimated by cross-validation using the multivariate normal likelihood function, although it can be specified via `shrink.param` as any value between 0 and 1 (0="I" and 1="R", values closer towards 0 indicate more shrinkage towards "I"). The validation groups are chosen by random assignment and so you may observe some slight variation in the estimated shrinkage parameter in repeat analyses. See `ridgeParamEst` for more details.

Rather than stopping after testing for multivariate effects, it is often of interest to find out which response variables express significant effects. Univariate statistics are required to answer this question, and these are reported if requested. Setting `p.uni="unadjusted"` returns the resampling-based univariate ANOVA P-values as well as the multivariate P-values, whereas `p.uni="adjusted"` returns adjusted ANOVA P-values (that have been adjusted for multiple testing), calculated using a step-down resampling algorithm as in Westfall & Young (1993, Algorithm 2.8). This method provides strong control of family-wise error rates, and makes use of resampling (using the method controlled by `resampling`) to ensure inferences take into account correlation between variables.

Value

An object of class `"anova.manylm"`. A list containing at least:

`p.uni`	the supplied argument.
`test`	the supplied argument.
`cor.type`	the supplied argument.
`resample`	the supplied argument.
`nBoot`	the supplied argument.
`calc.rss`	the supplied argument.
`table`	the data frame containing the anova table.
`shrink.param`	the supplied argument.
`n.bootsdone`	the number of bootstrapping iterations that were done, i.e. had no error.
`n.iter.sing`	the number of bootstrap iterations where the resampled design matrix was singular and could only be used partly.
`one`	logical. whether the anova table was calculated for one `manylm` object or for several `manylm` objects.

If `p.uni="adjusted"` or `p.uni="unadjusted"` the output list also contains:

`uni.test`	a table showing the test statistics of the univariate tests

`uni.p` a table showing the p-values of the univariate tests

The print method for `anova.manylm` objects prints the output in a 'pretty' form.

Author(s)

Yi Wang, Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

References

Anderson, M.J. and J. Robinson (2001). Permutation tests for linear models. *Australian and New Zealand Journal of Statistics* 43, 75-88.

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and their Application.* Cambridge University Press, Cambridge.

Warton D.I. (2008). Penalized normal likelihood and ridge regularization of correlation and covariance matrices. *Journal of the American Statistical Association* 103, 340-349.

Warton D.I. and Hudson H.M. (2004). A MANOVA statistic is just as powerful as distance-based statistics, for multivariate abundances. *Ecology* 85(3), 858-874.

Westfall, P. H. and Young, S. S. (1993) *Resampling-based multiple testing.* John Wiley & Sons, New York.

Wu, C. F. J. (1986) Jackknife, Bootstrap and Other Resampling Methods in Regression Analysis. *The Annals of Statistics* 14:4, 1261-1295.

See Also

`manylm`, `summary.manylm`, `plot.manylm`

Examples

```
## Load the spider dataset:
data(spider)
spiddat <- log(spider$abund+1)
spiddat <- mvabund(spiddat)
spidx <- spider$x

## Fit several multivariate linear models:
fit <- manylm( spiddat ~ spidx ) # model with all explanatory variables

## Use the default residual resampling to test the significance of this model:
## return summary of the manylm model
anova(fit)

## intercept model
fit0 <- manylm(spiddat ~ 1)
## include soil and leaf variables
fit1 <- update(fit0, . ~ . + spidx[, c(1, 3)])
## include moss variables
fit2 <- update(fit1, . ~ . + spidx[, 4])

## Use (residual) resampling to test the significance of these models,
```

```
## accounting for correlation between variables by shrinking
## the correlation matrix to improve its stability:
anova(fit, fit0, fit1, fit2, cor.type="shrink")

## Use the sum of F statistics to estimate multivariate significance from
## 4999 resamples, and also reporting univariate statistics with
## adjusted P-values:
anova(fit, fit0, fit1, fit2, nBoot=4999, test="F", p.uni="adjust")
```

`anova.traitglm` *Testing for a environment-by-trait (fourth corner) interaction by analysis of deviance*

Description

Returns an analysis of deviance from a fourth corner model, as computed using `traitglm`, typically to test for an environment-by-trait interaction. Slowly! This function works via `anova.manyglm`, which uses row-resampling for inference, and it only applies to `traitglm` objects that have been fitted using the (default) `manyglm` function.

Usage

```
## S3 method for class 'traitglm'
anova(object, ..., nBoot=99, resamp="pit.trap", test="LR",
                            block = NULL, show.time="all", bootID=NULL)
```

Arguments

`object`	A fitted object of class `traitglm` and class `manyglm`.
`...`	Additional `traitglm` objects, fitted using the `formula` argument.
`nBoot`	The number of bootstrap iterations. Default is 99 (NOTE: you should increase this for later runs!)
`resamp,test,bootID`	Arguments as in `anova.manyglm`, to control resampling method (`resamp`), test statistic (`test`) and whether or not a matrix of bootstrap resamples is manually entered (`bootID`).
`block`	A factor specifying the sampling level to be resampled. Default is resampling sites (which still involves passing a blocking variable to `anova.manyglm`, to keep all rows of the original abundance data together in resamples).
`show.time`	Whether to display timing information for the resampling procedure: this is advisable, as resampling fourth corner models many times can take a while. The default value `"all"` shows all timing information, `"total"` shows only the overall time taken for the tests, and `"none"` shows none.

Details

There are two possible uses of this function, depending whether one traitglm object is specified or multiple objects.

If one traitglm object is specified, the anova.traitglm function returns a table summarising the statistical significance of the fourth corner terms in a model, that is, the interaction between environment and traits in predicting abundance across taxa and sites. All environment-by-trait interaction terms from the model are simultaneously tested.

If two or more nested traitglm objects are specified, and each has been fitted using a formula argument to the same set of datasets, then sequential test statistics (and P values) are returned for each additional model fit.

All traitglm models must be fitted using the manyglm function, which is its default behaviour, in order to access the anova.manyglm, which does most of the work. See anova.manyglm for details on how resampling is done, and options for arguments controlling the test statistic (via test) and the resampling method (via resamp). Because traitglm models are fitted by first vectorising the data into a univariate model, arguments such as p.uni and cor.type are redundant.

traitglm fits a single model to abundances across all sites and taxa at the same time, meaning the vector of abundances is typically pretty long, and the design matrix explaining how abundance varies across taxa and sites is typically pretty large. So resampling can take yonks. Hence the default number of resamples has been set at nBoot=99, but please consider increasing this once you have a sense for how long it will take to run (scales roughly linearly with nBoot).

Value

A list of values as returned by anova.manyglm, of which the most relevant element is table (the analysis of deviance table).

Warning

The comparison between two or more models by anova.traitglm will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values and R's default of na.action = na.omit is used.

Author(s)

David I. Warton <David.Warton@unsw.edu.au>

References

Warton DI, Shipley B & Hastie T (2015) CATS regression - a model-based approach to studying trait-based community assembly, Methods in Ecology and Evolution 6, 389-398.

See Also

anova.manyglm,traitglm

Examples

```
data(antTraits)

# we'll fit a small fourth corner model, to a subset of the antTraits data.
# first select only species present in at least 25% of plots:
abSum = apply(antTraits$abund>0,2,mean)
ab = antTraits$abund[,abSum>0.25]
tr = antTraits$traits[abSum>0.25,]

# now fit the fourth corner model, only as a function of a couple of traits and env variables:
ft=traitglm(ab,antTraits$env[,1:3],data.frame(tr$Weber,tr$Femur))
anova(ft,nBoot=39)
# Note you should refit with more bootstrap samples (e.g. 999), should take <2 minutes to run

# for an example using anova.traitglm for multiple fits, uncomment the following lines:
# ft2=traitglm(antTraits$abund,antTraits$env[,3:4],antTraits$traits[,c(1,3)],
#   formula=~1,composition=TRUE) #no fourth corner terms
# ft3=traitglm(antTraits$abund,antTraits$env[,3:4],antTraits$traits[,c(1,3)],
#  formula=~Shrub.cover:Femur.length+Shrub.cover:Pilosity,composition=TRUE) #shrub interactions
# ft4=traitglm(antTraits$abund,antTraits$env[,3:4],antTraits$traits[,c(1,3)],
#  formula=~Shrub.cover:Femur.length+Shrub.cover:Pilosity+Volume.lying.CWD:Femur.length+
#   Volume.lying.CWD:Pilosity, composition=TRUE) #all interactions only
# anova(ft2,ft3,ft4) # Shrub interactions not significant but CWD interactions are.
```

---

antTraits *Ant data, with species traits*

Description

Abundances of 41 epigaeic ant species across 30 sites in south-eastern Australia, with species trait and environmental data

Usage

```
data(antTraits)
```

Format

A list containing three elements:

abund A data frame with observations at 30 different locations of abundances of 41 epigaeic ant species.

env A data frame containing 7 environmental variables from transects at each of the 30 sites:

Bare.ground Percent cover of bare ground, as estimated from ten 1x1 metre quadrats

Canopy.cover Percent canopy cover, as estimated from two 20x20m transects

Shrub.cover Percent canopy cover, as estimated from two 20x20m transects

Volume.lying.CWD Estimated volume of Coarse Woody Debris in two 20x20m transects, including all debris >5cm diameter.

Feral.mammal.dung Proportion of quadrats including mammal dung, out of ten 1x1m quadrats.

traits A data frame containing 5 species traits measured for each of the 41 species. Weber's length was log-transformed, Femur length was log-transformed then regressed against log(Weber's length), to remove the effect of size.

Femur.length Residuals from regression of log(Femur length) against log(Weber's length)

No.spines Number of spines on propodeum and petioles, as an integer value

Pilosity A factor with four levels of pilosity, 0 = No or very few hairs; 1 = a sparse but regular covering of hairs; 2 = a consistent, moderate covering of hair; 3 = very dense hair covering

Polymorphism 0 = Monomorphic, 1 = polymorphic, 2 = dimorphic

Webers.length log transformed. Body length, measured as the distance from the anterodorsal margin of the pronotum to the posteroventral margin of the propodeum

References

Gibb H, Stoklosa J, Warton, DI, Brown, AM, Andrew, NR and Cunningham, SA (2015) Does morphology predict trophic position and habitat use of ant species and assemblages? Oecologia 177, 519-531.

Examples

```
data(antTraits)
ft = traitglm(antTraits$abund,antTraits$env,antTraits$traits) #to do a fourth corner analysis
```

`best.r.sq` *Use R^2 to find the variables that best explain a multivariate response.*

Description

Finds the subset of explanatory variables in a formula that best explain the variation in a multivariate response, as measured by a chosen definition of R^2. Modifications are included for high dimensional data, such as multivariate abundance data in ecology.

Usage

```
best.r.sq(formula, data = parent.frame(), subset, var.subset,
  n.xvars= min(3, length(xn)), R2="h", ...)
```

Arguments

`formula`	a mvformula, a multivariate formula.
`data`	optional, the data.frame (or list) from which the variables in formula should be taken.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`var.subset`	an optional vector specifying the subset of the responses to be used.

n.xvars	the number of independent variables with the highest average R^2 that should be found.
R2	the type of R^2 (correlation coefficient) that should be shown, possible values are: "h" = Hooper's R^2 = tr(SST^(-1)SSR))/p "v" = vector R^2 = det(SSR)/det(SST) "n" = none Note that for a univariate response, all of these are equivalent to the ordinary product-moment correlation coefficient.
...	further arguments that are passed on to lm.

Details

best.r.sq finds the n.xvars influence variables obtained by a forward selection in a multivariate linear model given by formula.
Only the response variables given by var.subset are considered. However, if var.subset is NULL all response variables are considered.
Interactions are excluded from the search mechanism, however the indices that are returned correspond to the indices in the model. This function is intended as an exploratory tool which can be used for example in plotting, and is not intended as a tool for formal model selection. choose 'all possible subsets' the moment)

Value

This function returns a list consisting of:

xs a vector of indices of independent variables with the greatest explanatory power, as previously.

r2Step a vector of total R^2 from sequential model fits including each of the model terms identified in xs.

r2Matrix a matrix containing the total R^2 for each term in the model at each addition step (steps in columns and model terms in rows).

Author(s)

Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

best.r.sq( spiddat~X )
```

boxplot.mvabund *Boxplots for multivariate abundance Data*

Description

Draw Boxplots of mvabund or mvformula Objects

Usage

```
## S3 method for class 'mvabund'
boxplot(x, y, range=1.5, names=NULL, at=NULL,
                 n.vars=min(12,NCOL(x)), overall.main="Boxplot",
                 var.subset=NA, transformation="log", ...)

## S3 method for class 'mvformula'
boxplot(
    x, n.vars=12, overall.main="", var.subset=NA, ...)
```

Arguments

x	for the mvabund method x specifies the data from which the boxplots are to be produced. This can be either a numeric vector, or a single list containing such vectors. Additional unnamed arguments specify further data as separate vectors (each corresponding to a component boxplot). NAs are allowed in the data. For the default method, unnamed arguments are additional data vectors (unless x is a list when they are ignored), and named arguments are arguments and graphical parameters to be passed to in addition to the ones given by argument pars (and override those in pars). For the mvformula method, a formula, such as y ~ grp, where y is a numeric mvabund object of data values to be split into groups according to the grouping variable grp (a factor).
y	for the mvabund method y can be an additional mvabund object, if x isn't a list.
range	this determines how far the plot whiskers extend out from the box. If range is positive, the whiskers extend to the most extreme data point which is no more than range times the interquartile range from the box. A value of zero causes the whiskers to extend to the data extremes.
names	only available for the mvabund method: group labels which will be printed under each boxplot.
at	only available for the mvabund method: numeric vector giving the locations where the boxplots should be drawn; defaults to 1:n where n is the number of boxes.
n.vars	the number of variables to include in the plot.
overall.main	a character to display as title for every window.
var.subset	a numeric vector of indices indicating which variables of the mvabund.object should be included on the plot.

`transformation` an optional transformation, (ONLY) for the `mvabund` method. Note, that for the `mvabund` method `transformation` must be used instead of `log`.
Available values are:
"no" = untransformed, "sqrt"=square root transformed, "log" (default)=log(Y/min+1) transformed, "sqrt4" =4th root transformed.

`...` for the `mvformula` method, named arguments to be passed to the `plot.mvformula` method. Some arguments that are available for the `mvabund` method, are not available in `plot.mvformula` and can therefore not available in the `mvformula` method.

For the `mvabund` method, unamed arguments are additional data of vectors or matrices or `mvabund` objects, (unless `x` is a list when they are ignored),and named arguments are arguments and graphical parameters to be passed in addition to the ones given by argument `pars` (and override those in `pars`).

Details

The function `boxplot.mvabund` allows simultaneous construction of many variables on a single figure. Thus a good comparative overview about the distribution of abundances for several species can be obtained.
There are several ways in which this function can be used. If one `mvabund` object, either named `x` or `y` or not names, is passed, it will be drawn on one plot and abundances can be compared over several variables.
If two `mvabund` objects, named `x` and `y` are passed for plotting, they will be shown on one plot, showing for each species the abundances of both objects directly one below the other.
If more than two `mvabund` objects are passed, each of them will be plotted separately.
Additionally, it is possible to specify `x` as a list of `mvabund` objects. Each of them will be plotted separately and any further `mvabund` data will be ignored, regardless if it is specified as `y` or unnamed.

The function `boxplot.mvformula` can be used to draw boxplots of a `mvabund` object in dependence of explanatory variables. The explanatory variables can be both numerical values as well as factor variables. If the formula contains both of them, there will be separate plots for the terms with numerical values and the terms with factor variables, displayed on separate windows.

The arguments `plot`, `varwidth` and `add`, which are availabe in the default method of `boxplot`, are not available for the `mvabund` and `mvformula` methods. The argument `horizontal` is not available for the `mvabund` method.
A number of other arguments like `at` and `names` are only available for the `mvabund` method.

Value

In contrast to the default method (boxplot.default) nothing will be returned. These functions are only used for drawing the plots.

Warning

The argument `log`, that is available in most plotting functions can not be used for plotting `mvabund` or `mvformula` objects. Instead use `transformation` for the `mvabund` method and for the `mvformula` method include transformations in the formula.

Author(s)

Ulrike Naumann, Yi Wang, Stephen Wright and David Warton <David.Warton@unsw.edu.au>.

References

Warton, D. I. ( ) *Raw data graphing: an informative but under-utilised tool for the analysis of multivariate abundances*, , .

See Also

plot.mvabund.

Examples

```
require(graphics)

#### Basic Use ####
data(spider)
spiddat <- spider$abund
X <- spider$x

## Create the mvabund object:
spiddat <- mvabund(spiddat)

## Draw a boxplot for a mvabund object:
boxplot(spiddat)

## the same plot could be done by
plot(spiddat,type="bx")


#### Advanced Use ####
data(solberg)
solbdat <- mvabund(solberg$abund)
treatment<- solberg$x

# create pch type and colour vectors
treat.pch <- treat.col <- unclass(treatment)

# Boxplot for data
plot.mvabund(x=solbdat,y=treatment,type="bx",
             main="BoxPlot of The 12 Highest Abundant Species",
             xlab="Abundance [sqrt scale]",ylab="",
             transformation="sqrt",t.lab="o",shift=TRUE)
```

`cv.glm1path` *Fits a path of Generalised Linear Models with LASSO (or L1) penalties, and finds the best model by corss-validation.*

Description

Fits a sequence (path) of generalised linear models with LASSO penalties, using an iteratively reweighted local linearisation approach. The whole path of models is returned, as well as the one that minimises predictive log-likelihood on random test observations. Can handle negative binomial family, even with overdispersion parameter unknown, as well as other GLM families.

Usage

```
cv.glm1path(object, block = NULL, best="min", plot=TRUE, prop.test=0.2, n.split = 10,
    seed=NULL, show.progress=FALSE, ...)
```

Arguments

`object`	Output from a `glm1path` fit.
`block`	A factor specifying a blocking variable, where training/test splits randomly assign blocks of observations to different groups rather than breaking up observations within blocks. Default (`NULL`) will randomly split rows into test and training groups.
`best`	How should the best-fitting model be determined? `"1se"` uses the one standard error rule, `"min"` (or any other value) will return the model with best predictive performance. WARNING: David needs to check se calculatios...
`plot`	Logical value indicating whether to plot the predictive log-likelihood as a function of model complexity.
`prop.test`	The proportion of observations (or blocks) to assign as test observations. Default value of 0.2 gives a 80:20 training:test split.
`n.split`	The number of random training/test splits to use. Default is 10 but the more the merrier (and the slower).
`seed`	A vector of seeds to use for the random test/training splits. This is useful if you want to be able to exactly replicate analyses, without Monte Carlo variation in the splits. Default will not used fixed seeds.
`show.progress`	Logical argument, if TRUE, console will report when a run for a seed has been completed. This option has been included because this function can take yonks to run on large datasets.
`...`	Further arguments passed through to `glm1path`.

Details

This function fits a series of LASSO-penalised generalised linear models, with different values for the LASSO penalty, as for glm1path. The main difference is that the best fitting model is selected by cross-validation, using n.test different random training/test splits to estimate predictive performance on new (test) data. Mean predictive log-likelihood (per test observation) is used as the criterion for choosing the best model, which has connections with the Kullback-Leibler distance. The best argument controls whether to select the model that maximises predictive log-likelihood, or the smallest model within 1se of the maximum (the '1 standard error rule').

All other details of this function are as for glm1path.

Value

coefficients	Vector of model coefficients for the best-fitting model (as judged by predictive log-likelihood)
lambda	The value of the LASOS penalty parameter, lambda, for the best-fitting model (as judged by predictive log-likelihood)
glm1.best	The glm1 fit for the best-fitting model (as judged by predictive log-likelihood). For what this contains see glm1.
all.coefficients	A matrix where each column represents the model coefficients for a fit along the path specified by lambdas.
lambdas	A vector specifying the path of values for the LASSO penalty, arranged from largest (strongest penalty, smallest fitted model) to smallest (giving the largest fitted model).
logL	A vector of log-likelihood values for each model along the path.
df	A vector giving the number of non-zero parameter estimates (a crude measure of degrees of freedom) for each model along the path.
bics	A vector of BIC values for each model along the path. Calculated using a penalty on model complexity as specified by input argument k.
counter	A vector counting how many iterations until convergence, for each model along the path.
check	A vector of logical values specifying whether or not Karush-Kuhn-Tucker conditions are satisfied at the solution.
phis	For negative binomial regression - a vector of overdispersion parameters, for each model along the path.
y	The vector of values for the response variable specified as an input argument.
X	The design matrix of p explanatory variables specified as an input argument.
penalty	The vector to be multiplied by each lambda to make the penalty for each fitted model.
family	The family argument specified as input.
ll.cv	The mean predictive log-likelihood, averaged over all observations and then over all training/test splits.
se	Estimated standard error of the mean predictive log-likelihood.

Author(s)

David I. Warton <David.Warton@unsw.edu.au>

References

Osborne, M.R., Presnell, B. and Turlach, B.A. (2000) On the LASSO and its dual. Journal of Computational and Graphical Statistics, 9, 319-337.

See Also

glm1path, \codeglm1, glm, family

Examples

```
data(spider)
Alopacce <- spider$abund[,1]
X <- cbind(1,spider$x)

# fit a LASSO-penalised negative binomial regression:
ft = glm1path(Alopacce,X,lam.min=0.1)
coef(ft)

# now estimate the best-fitting model by cross-validation:
cvft = cv.glm1path(ft)
coef(cvft)
```

---

deviance.manylm *Model Deviance*

Description

Returns the deviance of a fitted multivariate model object for abundance data.

Usage

```
## S3 method for class 'manylm'
deviance(object, na.action="na.omit", ...)
```

Arguments

object	the manylm object.
na.action	how to deal with NA values. Can be one of "na.omit", "na.exclude", "na.fail", NULL
...	additional optional arguments.

Value

The value of the deviance extracted from the object `object`.

See Also

`manylm`.

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

## Calculate the deviance:
deviance(manylm(spiddat~X))
```

`extend.x.formula` *Extend a Formula to all of it's Terms*

Description

extend a compact formula to all of it's terms as they are interpreted

Usage

```
extend.x.formula(formula, extend.term=TRUE, return.interaction=TRUE)
```

Arguments

`formula`	a model formula.
`extend.term`	logical. If `TRUE` terms that refer to multiple variables are split into it's multiple terms.
`return.interaction`	logical. Whether a list containing the new formula and a vector containing logical values with information about interactions should be returned or only the new formula.

Value

If `return.interaction` is `TRUE` a list containing the components:

`formula`	the new formula.
`is.interaction`	logical, vector giving information whether the corresponding formula term is an interaction or not.

Author(s)

Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

See Also

mvformula, formulaUnimva, plot.mvformula, best.r.sq.

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

foo <- mvformula(spiddat~ X[,1]*X[,2]+log(X[,3]))
extend.x.formula(foo)
```

---

formulaUnimva *Create a List of Univariate Formulas*

Description

Create a list of m univariate formulas given a formula with multivariate response of dimension m.

Usage

```
formulaUnimva(formula, var.subset, split.x=FALSE, intercept=0,
  allow.noresp=FALSE)
```

Arguments

formula	a formula or a mvformula, the elements are not allowed to be data.frames.
var.subset	optional vector of the variable numbers to use.
split.x	logical, whether explanatory terms that are matrices should be split and each added as a single term. this is useful for plotting formulas.
intercept	numeric, either 1 if an Intercept should be included in the formula or 0 if there shouldn't be an Intercept in the formula.
allow.noresp	logical, whether an empty response is allowed (a list with one element would be returned) or not (would result in an error.)

Value

A list containing m formulas with the univariate responses chosen by var.subset.

Author(s)

Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

See Also

mvformula, mvabund.

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

formulaUnimva(spiddat~X)
```

`glm1` *Fits a Generalised Linear Models with a LASSO (or L1) penalty, given a value of the penalty parameter.*

Description

Fits a generalised linear model with a LASSO penalty, using an iteratively reweighted local linearisation approach, given a value of the penalty parameter (lamb). Can handle negative binomial family, even with overdispersion parameter unknown, as well as other GLM families.

Usage

```
glm1(y, X, lambda, family = "negative.binomial", weights = rep(1, length(y)),
    b.init = NA, phi.init = NA, phi.method = "ML", tol = c(1e-08, .Machine$double.eps),
     n.iter = 100, phi.iter = 1)
```

Arguments

`y`	A vector of values for the response variable.
`X`	A design matrix of p explanatory variables.
`family`	The family of the response variable, see `family`. Negative binomial with unknown overdispersion can be specified as "negative.binomial", and is the default.
`lambda`	The penalty parameter applied to slope parameters. Different penalties can be specified for different parameters by specifying lamb as a vector, whose length is the number of columns of X. If scalar, this penalty is applied uniformly across all parameters except for the first (assuming that it is an intercept)
`weights`	Observation weights. These might be useful if you want to fit a Poisson point process model...
`b.init`	Initial slope estimate. Must be a vector of the same length as the number of columns in X.
`phi.init`	Initial estimate of the negative binomial overdispersion parameter. Must be scalar.
`phi.method`	Method of estimating overdispersion.
`tol`	A vector of two values, specifying convergence tolerance, and the value to truncate fitted values at.
`n.iter`	Number of iterations to attempt before bailing.

`phi.iter` Number of iterations estimating the negative binomial overdispersion parameter (if applicable) before returning to slope estimation. Default is one step, i.e. iterating between one-step estimates of beta and phi.

Details

This function fits a generalised linear model with a LASSO penalty, sometimes referred to as an L1 penalty or L1 norm, hence the name glm1. The model is fit using a local linearisation approach as in Osborne et al (2000), nested inside iteratively reweighted (penalised) least squares. Look it's not the fastest thing going around, try `glmnet` if you want something faster (and possibly rougher as an approximation). The main advantage of the `glm1` function is that it has been written to accept any glm family argument (although not yet tested beyond discrete data!), and also the negative binomial distribution, which is especially useful for modelling overdispersed counts.

For negative binomial with unknown overdispersion use `"negative.binomial"`, or if overdispersion is to be specified, use `negative.binomial(theta)` as in the `MASS` package. Note that the output refers to phi=1/theta, i.e. the overdispersion is parameterised such that the variance is mu+phi*mu^2. Hence values of phi close to zero suggest little overdispersion, values over one suggest a lot.

Value

`coefficients`	Vector of parameter estimates
`fitted.values`	Vector of predicted values (on scale of the original response)
`logLs`	Vector of log-likelihoods at each iteration of the model. The last entry is the log-likelihood for the final fit.
`phis`	Estimated overdispersion parameter at each iteration, for a negative binomial fit.
`phi`	Final estimate of the overdispersion parameter, for a negative binomial fit.
`score`	Vector of score equation values for each parameter in the model.
`counter`	Number of iterations until convergence. Set to Inf for a model that didn't converge.
`check`	Logical for whether the Kuhn-KArush-Tucker conditions are saitsfied.

Author(s)

David I. Warton <David.Warton@unsw.edu.au>, Ian W. Renner and Luke Wilson.

References

Osborne, M.R., Presnell, B. and Turlach, B.A. (2000) On the LASSO and its dual. Journal of Computational and Graphical Statistics, 9, 319-337.

See Also

`glm1path`, `glm1`, `glm`, `family`

Examples

```
data(spider)
Alopacce <- spider$abund[,1]
X <- cbind(1,spider$x)
#fit a LASSO-penalised negative binomial GLM, with penalty parameter 10:
ft = glm1(Alopacce,X,lambda=10)

plot(ft$logLs) # a plot of the log-likelihood, each iteration to convergence
coef(ft) # coefficients in the final model
```

`glm1path` *Fits a path of Generalised Linear Models with LASSO (or L1) penalties, and finds the model that minimises BIC.*

Description

Fits a sequence (path) of generalised linear models with LASSO penalties, using an iteratively reweighted local linearisation approach. The whole path of models is returned, as well as the one that minimises BIC. Can handle negative binomial family, even with overdispersion parameter unknown, as well as other GLM families.

Usage

```
glm1path(y, X, family = "negative.binomial", lambdas = NULL,
  penalty = c(0, rep(1, dim(X)[2]-1)), df.max = sum(y > 0), n.lambda = 25, lam.max = NULL,
  lam.min = NULL, k = log(length(y)), b.init = NA, phi.init = NA, phi.iter = 1, ...)
```

Arguments

`y`	A vector of values for the response variable.
`X`	A design matrix of p explanatory variables.
`family`	The family of the response variable, see `family`. Negative binomial with unknown overdispersion can be specified as "negative.binomial", and is the default.
`lambdas`	An optional vector of LASSO penalty parameters, specifying the path along which models will be fitted. This penalty is applied to parameters as specified in `penalty`. By default, a geometric sequence of values will be constructed with `n.lambda` values, starting from the intercept model and reducing lambda to 1.e-6 of its original value. Any vector that is provided will be sorted in decreasing order, so that the smallest model (biggest penalty) is fitted first.
`penalty`	A vector to be multiplied by each lambda to make the penalty for each fitted model. The main purpose here is to allow penalties to be applied to some parameters but not others, but it could also be used to change the size of the penalty for some terms as compared to others (e.g. to fit an adaptive LASSO). Must have the same length as the dimension of the model, `dim(X)[2]`.

df.max	The maximum number of terms that is permitted in the fitted model. Once this threshhold is reached no further fits are attempted. The default break-point is the number of non-zero values in the response vector.
n.lambda	The number of models to fit along the path (if not previously specified via lambdas).
lam.max	The maximum value of the LASSO penalty to use along the path of fitted values (if not previously specified via lambdas).
lam.min	The minimum value of the LASSO penalty to use along the path of fitted values (if not previously specified via lambdas).
k	In BIC calculation, this is the value of the penalty per parameter in the fitted model. The default value, log(length(y)), gives BIC (known to be consistent, for adaptive LASSO), changing it to 2 would give AIC (which is not so great in terms of properties).
b.init	An initial value for beta for the first model along the fitted path. Default is to fit an intercept model.
phi.init	For negative binomial models: An initial value for the overdispersion parameter for the first model along the fitted path. Default is zero (Poisson fit).
phi.iter	Number of iterations estimating the negative binomial overdispersion parameter (if applicable) before returning to slope estimation. Default is one step, i.e. iterating between one-step estimates of beta and phi.
...	Arguments passed to glm1.

Details

This function fits a series of LASSO-penalised generalised linear models, with different values for the LASSO penalty. Largely inspired by the glmnet package. This results in a path of fitted models, from small ones (with big LASSO penalties) to larger ones (with smaller penalties). Each individual model is fitted using the glm1 function, which uses a local linearisation approach as in Osborne et al (2000), nested inside iteratively reweighted (penalised) least squares, and using results from the previous fit as initial estimates. Look it's not the fastest thing going around, try glmnet if you want something faster (and possibly rougher as an approximation). The main advantage of the glm1path function is that it has been written to accept any glm family argument (although not yet tested beyond discrete data!), and also the negative binomial distribution, which is especially useful for modelling overdispersed counts.

For negative binomial with unknown overdispersion use "negative.binomial", or if overdispersion is to be specified, use negative.binomial(theta) as in the MASS package. Note that the output refers to phi=1/theta, i.e. the overdispersion is parameterised in output such that the variance is mu+phi*mu^2. Hence values of phi close to zero suggest little overdispersion, values over one suggest a lot.

You can use the residuals and plot functions on glm1path objects in order to compute Dunn-Smyth residuals and a plots of these residuals against linear predictors, as for manyglm.

Value

An object of class glm1path with the following components:

coefficients Vector of model coefficients for the best-fitting model (as judged by BIC)

lambda The value of the LASOS penalty parameter, lambda, for the best-fitting model (as judged by BIC)

glm1.best The glm1 fit for the best-fitting model (as judged by BIC). For what this contains see `glm1`.

all.coefficients A matrix where each column represents the model coefficients for a fit along the path specified by `lambdas`.

lambdas A vector specifying the path of values for the LASSO penalty, arranged from largest (strongest penalty, smallest fitted model) to smallest (giving the largest fitted model).

logL A vector of log-likelihood values for each model along the path.

df A vector giving the number of non-zero parameter estimates (a crude measure of degrees of freedom) for each model along the path.

bics A vector of BIC values for each model along the path. Calculated using a penalty on model complexity as specified by input argument `k`.

counter A vector counting how many iterations until convergence, for each model along the path.

check A vector of logical values specifying whether or not Karush-Kuhn-Tucker conditions are satisfied at the solution.

phis For negative binomial regression - a vector of overdispersion parameters, for each model along the path.

y The vector of values for the response variable specified as an input argument.

X The design matrix of p explanatory variables specified as an input argument.

penalty The vector to be multiplied by each lambda to make the penalty for each fitted model.

family The family argument specified as input.

Author(s)

David I. Warton <David.Warton@unsw.edu.au>

References

Osborne, M.R., Presnell, B. and Turlach, B.A. (2000) On the LASSO and its dual. Journal of Computational and Graphical Statistics, 9, 319-337.

See Also

`glm1`, `glm`, `family`, `residuals.manyglm`, `plot.manyany`

Examples

```
data(spider)
Alopacce <- spider$abund[,1]
X <- cbind(1,spider$x)

# fit a LASSO-penalised negative binomial regression:
ft = glm1path(Alopacce,X)
# have a look at the BICS for all models:
```

```
plot(ft$bics~ft$lambdas, log="x")

#the action seems to be at lambda above 0.1, re-do with a minimum lambda at 0.1 and more lambdas:
ft2 = glm1path(Alopacce,X,lam.min=0.1,n.lambda=100)
plot(ft2$bics~ft2$lambdas, log="x")

# return the slope estimates for the best-fitting model:
coef(ft2)

# look at a residual plot:
plot(ft2)
```

`logLik.manylm` *Calculate the Log Likelihood*

Description

Calculate the log likelihood of a multivariate linear model.

Usage

```
## S3 method for class 'manylm'
logLik(object, REML = FALSE, ...)
```

Arguments

`object`	a `manylm` object from which a log-likelihood value should be extracted.
`...`	some methods for this function require additional arguments.
`REML`	an optional logical value. If `TRUE` the restricted log-likelihood is returned, else, if `FALSE`, the log-likelihood is returned. Defaults to `FALSE`.

Details

It is assumed that the scale has been estimated (by maximum likelihood or REML), and all the constants in the log-likelihood are included.

Value

Returns an object, say `r`, of class `logLik` which is a number with attributes, `attr(r, "df")` (**d**egrees of **f**reedom) giving the number of (estimated) parameters in the model.

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

lm.spider <- manylm(spiddat~X)
logLik(lm.spider)
```

---

manyany *Fitting Many Univariate Models to Multivariate Abundance Data*

Description

manyany is used to fit many univariate models (GLMs, GAMs, otherwise) to high-dimensional data, such as multivariate abundance data in ecology. This is the base model-fitting function - see plot.manyany for assumption checking, and anova.manyany for significance testing.

Usage

```
manyany(fn, yMat, formula, data, family="negative.binomial", composition = FALSE,
block = NULL, get.what="details", var.power=NA, na.action = "na.exclude", ...)
## S3 method for class 'manyany'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

fn	a character string giving the name of the function for the univariate model to be applied. e.g. "glm".
yMat	a matrix of response variables, e.g. multivariate abundances.
formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
data	a data frame containing predictor variables (a matrix is also acceptable). This is REQUIRED and needs to have more than one variable in it (even if only one is used in the model).
family	a description of the error distribution function to be used in the model, either as a character string, a family object, or a list of such objects, one for each response variable in the dataset. Such a list enables the fitting of models with different distributions for different responses. See Details for the families currently supported.
composition	logical. FALSE (default) fits a separate model to each species. TRUE fits a single model to all variables, including site as a row effect, such that all other terms model relative abundance (compositional effects).
block	a factor specifying the sampling level to be resampled. Default is resampling rows (if composition=TRUE in the manyany command, this means resampling rows of data as originally sent to manyany).

get.what	what to return from each model fit: "details" (default) includes predicted values and residuals in output, "models" also returns the fitted objects for each model, "none" returns just the log-likelihood (mostly for internal use).
var.power	the power parameter, if using the tweedie distribution.
na.action	Default set to exclude (for details see na.exclude) to avoid errors when NA's in predictors.
...	further arguments passed to the fitting function.
x	an object of class "manyany", usually, a result of a call to manyany.
digits	how many digits to include in the printed anova table.

Details

manyany can be used to fit the specified model type to many variables simultaneously, a generalisation of manyglm. It should be able to handle any fixed effects modelling function that has predict and logLik functions, and that accepts a family argument, provided that the family is on our list (currently 'gaussian', 'poisson', 'binomial', 'negative.binomial' and 'tweedie', although models for ordinal data are also accepted if using the clm function form the ordinal package). Models for manyany are specified symbolically, see for example the details section of lm and formula.

Unlike manyglm, this function accepts family functions as arguments instead of just character strings, giving greater flexibility. For example, you can use family=binomial(link="cloglog") to fit a model using the complementary log-log link, rather than being restricted to the default logit link.

A data argument is required, and it must be a dataframe containing more than one object. It need not contain that matrix of response variables, that is specified separately as yMat.

Setting composition=TRUE enables compositional analyses, where predictors are used to model relative abundance rather than mean abundance. This is achieved by vectorising the response matrix and fitting a single model across all variables, with a row effect to account for differences in relative abundance across rows. The default composition=FALSE just fits a separate model for each variable.

Value

manyany returns an object inheriting from "manyany".

The function anova (i.e. anova.manyany) will produce a significance test comparing two manyany objects. Currently there is no summary resampling function for objects of this class.

The generic accessor functions fitted.values, residuals, logLik, AIC, plot can be used to extract various useful features of the value returned by manyany.

An object of class "manyany" is a list containing at least the following components:

logL	a vector of log-likelihood terms for each response variable in the fitted model.
fitted.values	the matrix of fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
residuals	the matrix of probability integral transform (PIT) residuals. If the fitted model is a good fit, these will be approximately standard uniformly distributed.

`linear.predictor`
: the linear fit on link scale. But for ordinal models fitted using `clm`, these values are for the first category only.

`family`
: a vector of `family` arguments, one for each response variable.

`call`
: the matched call.

`model`
: the `model.frame` from the model for the last response variable.

`terms`
: a list of `terms` from the model for the last response variable.

Author(s)

David Warton <David.Warton@unsw.edu.au>.

References

Warton D. I., Wright S., and Wang, Y. (2012). Distance-based multivariate analyses confound location and dispersion effects. *Methods in Ecology and Evolution*, 3(1), 89-101.

See Also

`anova.manyany`, `residuals.manyany`, `plot.manyany`.

Examples

```
data(spider)
abund <- spider$abund
X <- as.matrix(spider$x)

## To fit a log-linear model assuming counts are negative binomial, via manyglm:
spidNB <- manyany("manyglm",abund,data=X,abund~X,family="negative.binomial")

logLik(spidNB) # a number of generic functions are applible to manyany objects

## To fit a glm with complementary log-log link to presence/absence data:
PAdat = pmin(as.matrix(abund),1) #constructing presence/absence dataset
spidPA <- manyany("glm",PAdat,data=X,PAdat~X,family=binomial("cloglog"))
plot(spidPA)
# There are some wild values in there for the Pardmont variable (residuals >5 or <-8).
#The Pardmont model didn't converge, coefficients are a bit crazy:
coef(spidPA)

# Can try again using the glm2 package to fit the models, this fixes things up:
# library(glm2)
# spidPA2 <- manyany("glm",PAdat,data=X,PAdat~X,family=binomial("cloglog"),method="glm.fit2")
# plot(spidPA2) #looks much better.

## To simultaneously fit models to ordinal data using the ordinal package:
# library(ordinal)
## First construct an ordinal dataset:
# spidOrd = abund
# spidOrd[abund>1 & abund<=10]=2
# spidOrd[abund>10]=3
```

```
# for(iVar in 1:dim(spidOrd)[2])
#   spidOrd[,iVar]=factor(spidOrd[,iVar])
##Now fit a model using the clm function:
# manyOrd=manyany("clm",spidOrd,abund~bare.sand+fallen.leaves,data=X)
# plot(manyOrd)
```

---

manyglm *Fitting Generalized Linear Models for Multivariate Abundance Data*

Description

manyglm is used to fit generalized linear models to high-dimensional data, such as multivariate abundance data in ecology. This is the base model-fitting function - see plot.manyglm for assumption checking, and anova.manyglm or summary.manyglm for significance testing.

Usage

```
manyglm(formula, family="negative.binomial", K=1, data=NULL, subset=NULL,
    na.action=options("na.action"), theta.method = "PHI", model = FALSE,
    x = TRUE, y = TRUE, qr = TRUE, cor.type= "I", shrink.param=NULL,
    tol=sqrt(.Machine$double.eps), maxiter=25, maxiter2=10,
    show.coef=FALSE, show.fitted=FALSE, show.residuals=FALSE,
    show.warning=FALSE, offset, ... )
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
family	a description of the distribution function to be used in the in the model. The default is negative binomial regression (using a log link, with unknown overdispersion parameter), the following family functions are also accepted: binomial(), binomial(link="cloglog"), poisson(), which can also be specified using character strings as 'binomial', 'cloglog' and 'poisson', respectively. In future we hope to include other family functions as described in family.
K	number of trials in binomial regression. By default, K=1 for presence-absence data using logistic regression.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.

`na.action` a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is NULL, no action. Value `na.exclude` can be useful.

`theta.method` the method used for the estimation of the overdisperson parameter theta, such that the mean-variance relationship is V=m+m^2/theta for the negative binomial family. Here offers three options
"PHI" = Maximum likelihood estimation with respect to phi (default)
"ML" = Maximum likelihood estimation with respect to theta, as in Lawless(1987),
"Chi2" = Moment estimation using chi-square dampening on the log scale, as in Hilbe(2008).

`model, x, y, qr`
logicals. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the model matrix, the response, the QR decomposition of the model matrix) are returned.

`cor.type` the structure imposed on the estimated correlation matrix under the fitted model. Can be "I"(default), "shrink", or "R". See Details. This parameter is merely stored in `manyglm`, and will be used as the default value for `cor.type` in subsequent functions for inference.

`shrink.param` shrinkage parameter to be used if `cor.type="shrink"`. If a numerical value is not supplied, it will be estimated from the data by cross validation-penalised normal likelihood as in Warton (2008). The parameter value is stored as an attribute of the `manyglm` object, and will be used in subsequent functions for inference.

`tol` the tolerance used in estimation.

`maxiter` maximum allowed iterations in the weighted least square estimation of beta. The default value is 25.

`maxiter2` maximum allowed iterations in the internal ML estimations of negative binomial regression. The default value is 10.

`show.coef, show.fitted, show.residuals, show.warning`
logical. Whether to show model coefficients, fitted values, standardized pearson residuals, or operation warnings.

`offset` this can be used to specify an _a priori_ known component to be included in the linear predictor during fitting. This should be 'NULL' or a numeric vector of length equal to NROW (i.e. number of observations) or a matrix of NROW times p (i.e. number of species).

`...` further arguments passed to or from other methods.

Details

`manyglm` is used to calculate the parameter estimates of generalised linear models fitted to each of many variables simultaneously as in Warton et. al. (2012) and Wang et.al.(2012). Models for `manyglm` are specified symbolically. For details on how to specify a formula see the details section of `lm` and `formula`.

Generalised linear models are designed for non-normal data for which a distribution can be specified that offers a reasonable model for data, as specified using the argument `family`. The `manyglm`

function currently handles count and binary data, and accepts either a character argument or a family argument for common choices of family. For binary (presence/absence) data, `family=binomial()` can be used for logistic regression (logit link, "logistic regression"), or the complementary log-log link can be used `family=binomial("cloglog")`, arguably a better choice for presence-absence data. Poisson regression `family=poisson()` can be used for counts that are not "overdispersed" (that is, if the variance is not larger than the mean), although for multivariate abundance data it has been shown that the negative binomial distribution (`family="negative.binomial"`) is usually a better choice (Warton 2005). In both cases, a log-link is used. If another link function or family is desired, this can be specified using the `manyany` function, which accepts regular `family` arguments.

In negative binomial regression, the overdispersion parameter (`theta`) is estimated separately for each variable from the data, as controlled by `theta.method` for negative binomial distributions. We iterate between updates of `theta` and generalised linear model updates for regression parameters, as many as `maxiter2` times.

`cor.type` is the structure imposed on the estimated correlation matrix under the fitted model. Possible values are:
`"I"`(default) = independence is assumed (correlation matrix is the identity)
`"shrink"` = sample correlation matrix is shrunk towards I to improve its stability.
`"R"` = unstructured correlation matrix is used. (Only available when N>p.)

If `cor.type=="shrink"`, a numerical value will be assigned to `shrink.param` either through the argument or by internal estimation. The working horse function for the internal estimation is `ridgeParamEst`, which is based on cross-validation (Warton 2008). The validation groups are chosen by random assignment, so some slight variation in the estimated values may be observed in repeat analyses. See `ridgeParamEst` for more details. The shrinkage parameter can be any value between 0 and 1 (0="I" and 1="R", values closer towards 0 indicate more shrinkage towards "I").

Value

`manyglm` returns an object inheriting from `"manyglm"`, `"manylm"` and "mglm".

The function `summary` (i.e. `summary.manyglm`) can be used to obtain or print a summary of the results and the function `anova` (i.e. `anova.manyglm`) to produce an analysis of variance table, although note that these functions use resampling so they can take a while to fit.

The generic accessor functions `coefficients`, `fitted.values` and `residuals` can be used to extract various useful features of the value returned by `manyglm`.

An object of class `"manyglm"` is a list containing at least the following components:

`coefficients` a named matrix of coefficients.

`var.coefficients`
the estimated variances of each coefficient.

`fitted.values` the matrix of fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.

`linear.predictor`
the linear fit on the scale of the linear predictor.

`residuals` the matrix of *working* residuals, that is the Pearson residuals standardized by the leverage adjustment h obtained from the diagonal elements of the hat matrix H.

`PIT.residuals` probability integral transform (PIT) residuals - for a model that fits well these should be approximately standard uniform values evenly scattered between 0

	and 1. Their calculation involves some randomisation, so different fits will return slightly different values for PIT residuals.
`sqrt.1_Hii`	the matrix of scale terms used to standardize the Pearson reidusals.
`var.estimator`	the estimated variance of each observation, computed using the corresponding family function.
`sqrt.weight`	the matrix of square root of *working* weights, estimated for the corresponding family function.
`theta`	the estimated nuisance parameters accounting for overdispersion
`two.loglike`	two times the log likelihood.
`deviance`	up to a constant, minus twice the maximized log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.
`iter`	the number of iterations of IWLS used.
`data`	a data frame storing the input data.
`stderr.coefficients`	the estimated standard error of each coefficient.
`phi`	the inverse of theta
`tol`	the tolerance used in estimations.
`maxiter,maxiter2,family,theta.method,cor.type,formula`	arguments supplied in the `manyglm` call.
`aic`	a vector returning Akaike's *An Information Criterion* for each response variable - minus twice the maximized log-likelihood plus twice the number of coefficients.
`AICsum`	the sum of the AIC's over all variables.
`shrink.param`	the shrink parameter to be used in subsequent inference.
`call`	the matched call.
`terms`	the `terms` object used.
`rank`	the numeric rank of the fitted linear model.
`xlevels`	(where relevant) a record of the levels of the factors used in fitting.
`df.residual`	the residual degrees of freedom.
`x`	if the argument `x` is `TRUE`, this is the design matrix used.
`y`	if the argument `y` is `TRUE`, this is the response variables used.
`model`	if the argument `model` is `TRUE`, this is the `model.frame`.
`qr`	if the argument `qr` is `TRUE`, this is the QR decomposition of the design matrix.
`show.coef,show.fitted,show.residuals`	arguments supplied in the `manyglm` call concerning what it presented in output.
`offset`	the `offset` data used (where applicable).

Author(s)

Yi Wang, Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

References

Lawless, J. F. (1987) *Negative binomial and mixed Poisson regression*, Canadian Journal of Statistics 15, 209-225.

Hilbe, J. M. (2008) *Negative Binomial Regression*, Cambridge University Press, Cambridge.

Warton D.I. (2005) *Many zeros does not mean zero inflation: comparing the goodness of-fit of parametric models to multivariate abundance data*, Environmetrics 16(3), 275-289.

Warton D.I. (2008). Penalized normal likelihood and ridge regularization of correlation and covariance matrices. *Journal of the American Statistical Association* 103, 340-349.

Warton D.I. (2011). Regularized sandwich estimators for analysis of high dimensional data using generalized estimating equations. *Biometrics*, 67(1), 116-123.

Warton D. I., Wright S., and Wang, Y. (2012). Distance-based multivariate analyses confound location and dispersion effects. *Methods in Ecology and Evolution*, 3(1), 89-101.

Wang Y., Neuman U., Wright S. and Warton D. I. (2012). mvabund: an R package for model-based analysis of multivariate abundance data. *Methods in Ecology and Evolution*. online 21 Feb 2012.

See Also

anova.manyglm, summary.manyglm, residuals.manyglm, plot.manyglm

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

#To fit a log-linear model assuming counts are poisson:
glm.spid <- manyglm(spiddat~X, family="poisson")
glm.spid

summary(glm.spid, resamp="residual")

#To fit a binomial regression model to presence/absence data:
pres.abs <- spiddat
pres.abs[pres.abs>0] = 1
X <- data.frame(spider$x) #turn into a data frame to refer to variables in formula
glm.spid.bin <- manyglm(pres.abs~soil.dry+bare.sand+moss, data=X, family="binomial")
glm.spid.bin
drop1(glm.spid.bin) #AICs for one-term deletions, suggests dropping bare.sand

glm2.spid.bin <- manyglm(pres.abs~soil.dry+moss, data=X, family="binomial")
drop1(glm2.spid.bin) #backward elimination suggests settling on this model.
```

manylm *Fitting Linear Models for Multivariate Abundance Data*

Description

manylm is used to fit multivariate linear models to high-dimensional data, such as multivariate abundance data in ecology.

This is the base model-fitting function - see plot.manylm for assumption checking, and anova.manylm or summary.manylm for significance testing.

Usage

```
manylm(
   formula, data=NULL,  subset=NULL, weights=NULL,
   na.action=options("na.action"),  method="qr", model=FALSE,
   x=TRUE, y=TRUE, qr=TRUE, singular.ok=TRUE, contrasts=NULL,
   offset, test="LR" , cor.type= "I", shrink.param=NULL,
   tol=1.0e-5, ...)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which manylm is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If non-null, weighted least squares is used with weights weights (that is, minimizing sum(weights*e^2)); otherwise ordinary least squares is used.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
method	the method to be used; for fitting, currently only method = "qr" is supported; method = "model.frame" returns the model frame (the same as with model = TRUE, see below).
model, x, y, qr	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
singular.ok	logical. If FALSE (the default in S but not in R) a singular fit is an error.

contrasts	an optional list. See the contrasts.arg of model.matrix.default.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length either one or equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if both are specified their sum is used. See model.offset.
test	choice of test statistic. Can be one of "LR" (default) = likelihood ratio statistic "F" = Lawley-Hotelling trace statistic NULL = no test This parameter is merely stored in manylm, and will be used as the default value of test in subsequent functions for inference.
cor.type	structure imposed on the estimated correlation matrix under the fitted model. Can be "I"(default), "shrink", or "R". See anova.manylm for details of its usage. This parameter will be used as the default value of cor.type in subsequent functions for inference.
shrink.param	shrinkage parameter to be used if cor.type="shrink". This parameter will be used as the default value of shrink.param in subsequent functions for inference.
tol	the tolerance used in estimations.
...	additional arguments to be passed to the low level regression fitting functions (see below).

Details

Models for manylm are specified symbolically. For details on this compare the details section of lm and formula. If the formula includes an offset, this is evaluated and subtracted from the response.
See model.matrix for some further details. The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula (see aov and demo(glm.vr) for an example).
A formula has an implied intercept term. To remove this use either y ~ x - 1 or y ~ 0 + x. See formula for more details of allowed formulae.
manylm calls the lower level function manylm.fit or manylm.wfit for the actual numerical computations. For programming only, you may consider doing likewise.
All of weights, subset and offset are evaluated in the same way as variables in formula, that is first in data and then in the environment of formula.
For details on arguments related to hypothesis testing (such as cor.type and resample) see summary.manylm or anova.manylm.

Value

manylm returns an object of c("manylm", "mlm", "lm") for multivariate formula response and of of class c("lm") for univariate response.

A manylm object is a list containing at least the following components:

coefficients	a named matrix of coefficients
residuals	the residuals matrix, that is response minus fitted values.

fitted.values	the matrix of the fitted mean values.
rank	the numeric rank of the fitted linear model.
weights	(only for weighted fits) the specified weights.
df.residual	the residual degrees of freedom.
hat.X	the hat matrix.
txX	the matrix (t(x)%*%x).
test	the test argument supplied.
cor.type	the cor.type argument supplied.
resample	the resample argument supplied.
nBoot	the nBoot argument supplied.
call	the matched call.
terms	the terms object used.
xlevels	(only where relevant) a record of the levels of the factors used in fitting.
model	if requested (the default), the model frame used.
offset	the offset used (missing if none were used).
y	if requested, the response matrix used.
x	if requested, the model matrix used.

In addition, non-null fits will have components assign and (unless not requested) qr relating to the linear fit, for use by extractor functions such as summary.manylm.

Author(s)

Yi Wang, Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

See Also

anova.manylm, summary.manylm, plot.manylm

Examples

```
data(spider)
spiddat <- log(spider$abund+1)
spiddat <- mvabund(spiddat)
X <- spider$x

lm.spider <- manylm(spiddat~X)
lm.spider

#Then use the plot function for diagnostic plots, and use anova or summary to
#evaluate significance of different model terms.
```

manylm.fit *workhose functions for fitting multivariate linear models*

Description

These are the workhorse functions called by manylm used to fit multivariate linear models. These should usually *not* be used directly unless by experienced users.

Usage

```
manylm.fit(x, y, offset = NULL, tol=1.0e-010, singular.ok = TRUE, ...)
manylm.wfit(x, y, w, offset = NULL, tol=1.0e-010, singular.ok = TRUE, ...)
```

Arguments

x	design matrix of dimension n * p.
y	matrix or an mvabund object of observations of dimension n*q.
w	vector of weights (length n) to be used in the fitting process for the manylm.wfit functions. Weighted least squares is used with weights w, i.e., sum(w * e^2) is minimized.
offset	numeric of length n). This can be used to specify an *a priori* known component to be included in the linear predictor during fitting.
tol	tolerance for the qr decomposition. Default is 1.0e-050.
singular.ok	logical. If FALSE, a singular model is an error.
...	currently disregarded.

Value

a list with components

coefficients	p vector
residuals	n vector or matrix
fitted.values	n vector or matrix
weights	n vector — *only* for the *wfit* functions.
rank	integer, giving the rank
qr	(not null fits) the QR decomposition.
df.residual	degrees of freedom of residuals
hat.X	the hat matrix.
txX	the matrix (t(x)%*%x).

Author(s)

Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

See Also

manylm

---

meanvar.plot *Construct Mean-Variance plots for Multivariate Abundance Data*

Description

Construct mean-variance plots, separately for each column of the input data, and separately for each level of any input factor that is given (via a formula). This function was specially written for high dimensional data where there are many correlated variables exhibiting a mean-variance structure, in particular, multivariate abundance data in ecology.

Usage

```
meanvar.plot(x, ...)

## S3 method for class 'mvabund'
meanvar.plot(
  x, n.vars=NULL, var.subset=NULL, subset=NULL, table=FALSE, ...)

## S3 method for class 'mvformula'
meanvar.plot(
  x, n.vars = NULL, var.subset=NULL, subset=NULL, table=FALSE,
  overall.main=NULL, overlay=TRUE, ...)
```

Arguments

x	an mvabund objects or a Model Formula (can be a formula or a mvformula) to be used.
n.vars	the number of variables to include in the plot.
var.subset	vector of indices indicating the variables to be included on the plot, (default: the n.vars most abundant variables).
subset	an optional vector specifying a subset of observations to be used.
table	logical, whether a table of the Means and Variances should be returned
overall.main	an overall title for the window.
overlay	logical, whether overall means/variances for all variables are calculated and drawn on a single plot or calculated and plotted separately for different variables.
...	arguments to be passed to or from other methods.

Details

meanvar.plot calculates a mean-variance plot for a dataset with many variables (e.g., Warton D. I., Wright S., and Wang, Y. (2012)).

The mean values and variances are calculated across all observations, unless a formula is given as the first argument which specifies a factor as the dependent variable. In this latter case the means and variances are calculated separately within the groups defined by these factors.

By default the means and variances of all variables (and all factor levels) are displayed on the same plot. If a formula is given and the explanatory variables contain factor variables, the mean values and variances for each factor level can be calculated and displayed either for all variables together or for each variable separately.

For the latter, set overlay to FALSE. The mean-variances corresponding to the different factors will be drawn in different colors, that can be chosen by specifying col. col can then either be a single color value (see par) with the number of values being at least the maximum number of levels of the factors. The same applies to pch.

If mfrow is NULL and mfcol is NULL, par("mfrow") is used. If all.labels = FALSE, only the x-axis labels at the bottom plot and the y-axis labels of plots on the right side of the window are printed if furthermore main=NULL only the graphics on the top contain the full title, the other ones an abreviated one.

Note, that if a log-transformation is used for displaying the data, a specific mean-variance relation will not be visible in the plot, if either the calculated mean is zero and log!="x" or log!="xy" or if the calculated variance is zero and log!="y" or log!="xy".

By default the y/x ratio of the axis, specified by asp, will be set to 1 if log!="xy". If the mean-variance relation is not displayed on a log scale and overlay is FALSE, it is most often not advisable to specify asp, as there might not be one choice of asp that is sensible for each of the plots.

Value

If table is TRUE a table of the Means and Variances is returned. Otherwise, only the plot(s) is/are drawn.

Author(s)

Ulrike Naumann, Stephen Wright and David Warton <David.Warton@unsw.edu.au>.

References

Warton D. I., Wright S., and Wang, Y. (2012). Distance-based multivariate analyses confound location and dispersion effects. *Methods in Ecology and Evolution*, 3(1), 89-101.

Warton D.I. (2008). Raw data graphing: an informative but under-utilized tool for the analysis of multivariate abundances. *Austral Ecology* 33(3), 290-300.

See Also

plot.mvabund plot.mvformula.

Examples

```
require(graphics)

## Load the tikus dataset:
data(tikus)
tikusdat <- mvabund(tikus$abund)
year <- tikus$x[,1]

## Plot mean-variance plot for a mvabund object with a log scale (default):
meanvar.plot(tikusdat)

## Again but without log-transformation of axes:
meanvar.plot(tikusdat,log="")

## A mean-variance plot, data organised by year,
## for 1981 and 1983 only, as in Figure 7a of Warton (2008):
is81or83 <- year==81 | year==83
meanvar.plot(tikusdat~year, subset=is81or83, col=c(1,10))
```

---

mvabund *Multivariate Abundance Data Objects*

Description

mvabund creates an mvabund object.
as.mvabund attempts to turn its argument into an mvabund object.
is.mvabund tests if the argument is an mvabund object.
mvabund is a class of objects for which special-purpose plotting and regression functions have been written in the mvabund-package. The above are useful preliminary functions before analysing data using the special-purpose functions. These new functions were written specially for the analysis of multivariate abundance data in ecology, hence the title 'mvabund'.

Usage

```
mvabund( ... , row.names=NULL, check.rows=FALSE, check.names=TRUE,
  var.names=NULL, neg=FALSE, na.rm=FALSE )

as.mvabund(x)

is.mvabund(x)
```

Arguments

...	these arguments are of either the form value or tag = value. Component names are created based on the tag (if present) or the deparsed argument itself.
row.names	NULL or a single integer or character string specifying a column to be used as row names, or a character or integer vector giving the row names for the mvabund object.

check.rows	if TRUE then the rows are checked for consistency of length and names.
check.names	logical. If TRUE then the names of the variables are checked to ensure that they are syntactically valid variable names and are not duplicated. If necessary they are adjusted (by make.names) so that they are.
var.names	NULL or a character vector giving the column names for the mvabund object.
neg	character. If FALSE negative values will cause an error message.
na.rm	logical, whether missing values should be removed.
x	an R object.

Details

It is desirable to convert abundance data to mvabund objects, to allow automatic use of all methods for mvabund objects, for example the provided methods for plotting, linear and generalised linear model-fitting and inference.

Some more technical details:
mvabund objects always have two dimensions.
mvabund converts its arguments into an mvabund object. The supplied argument can be a matrix, data frame, a list of vectors, or several vectors as separate arguments.

If elements of the created mvabund object are not numeric, a warning will be printed.

If row.names are not supplied, the row names of the mvabund object will be NULL. If the length of row.names does not match the number of rows or there are duplicates, an error message will result.

Value

mvabund and as.mvabund returns an mvabund object.
is.mvabund returns TRUE if x is a matrix and FALSE otherwise.

Author(s)

Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

See Also

unabund, mvformula, plot.mvabund, Also see the mvabund-package.

Examples

```
data(solberg)

## Create an mvabund object:
solbergdat <- mvabund(solberg$abund)

## Turn solberg$abund into an mvabund object and store as solbergdat:
solbergdat <- as.mvabund(solberg$abund)

## Check if solbergdat  is an mvabund object:
is.mvabund(solbergdat)
```

mvformula *Model Formulae for Multivariate Abundance Data*

Description

mvformula is a method to create an object of class mvformula
as.mvformula is a function to turn a formula into a mvformula
is.mvformula tests if its argument is a mvformula object. mvformula is a class of objects for which special-purpose plotting and regression functions have been written in the mvabund-package. The above are useful preliminary functions before analysing data using the special-purpose functions. These new functions were written specially for the analysis of multivariate abundance data in ecology, hence the title 'mvabund'.

Usage

```
mvformula(x)

as.mvformula(x)
is.mvformula(x)
```

Arguments

x	an R object.

Details

mvformula is a method to create an object of class mvformula If the response of the resulting formula is not a mvabund object, a warning will be printed. as.mvformula is a function to turn a formula into a mvformula if the response in x is a data.frame or an unsuitable object the conversion will fail.

Value

a formula mvabund for mvformula and as.mvformula a logical value indicating whether x is a mvformula object

Author(s)

Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

See Also

mvabund. manylm.

Examples

```
require(graphics)

data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

## Create a formula for multivariate abundance data:
foo <- mvformula( spiddat~X )

## Check whether foo is a mvformula:
is.mvformula(foo)

## Plot a mvformula:
plot(foo)
```

`plot.manyany` *Plot Diagnostics for a manyany or glm1path Object*

Description

A residual vs fits plot from a `manyany` or `glm1path` object.

Usage

```
## S3 method for class 'manyany'
plot( x, ...)
```

Arguments

`x`	manyany object, resulting from a call to `manyany`.
`...`	other parameters to be passed through to plotting functions.

Details

`plot.manyany` is used to check assumptions that are made when fitting a model via `manyany` or `glm1path`. As in Wang et al (2012), you should check the residual vs fits plot for no pattern (hence no suggestion of failure of any linearity and mean-variance assumptions). It is also desirable that residuals follow a straight line of slope one on a normal Q-Q plot.

These plots use Dunn-Smyth residuals (Dunn & Smyth 1996), described at `residuals.manyglm`. Note that for discrete data, these residuals involve random number generation, and will not return identical results on replicate runs - so it is recommended that you plot your data a few times to check if any pattern shows up consistently across replicate plots.

Note also that for `glm1path` objects, slope coefficients have been shrunk towards zero so it is not unusual to see an increasing slope on the residual plot, with larger residuals coinciding with larger fitted values. This arises a a consequent of shrinkage, check if it goes away upon removing the penaly term (e.g. on refitting using `manyglm`) before ringing any alarm bells.

Author(s)

David Warton <David.Warton@unsw.edu.au>.

References

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

Wang Y., Naumann U., Wright S.T. & Warton D.I. (2012). mvabund - an R package for model-based analysis of multivariate abundance data. Methods in Ecology and Evolution 3, 471-474.

See Also

glm1path, manyany, manyglm

Examples

```
require(graphics)

data(spider)
abund <- mvabund(spider$abund)
X <- spider$x

## Plot the diagnostics for a log-linear model assuming counts are poisson:
spidPois <- manyany("glm", abund, data=X, abund ~ X, family=poisson())
plot(spidPois,pch=19,cex=0.2)
## Fan-shape means trouble for our Poisson assumption.

## Try a negative binomial instead...
require(MASS) # this package is needed for its negative binomial family function
spidNB <- manyany("manyglm", abund, data=X, abund ~ X, family="negative.binomial")
plot(spidNB,pch=19,cex=0.2,xlim=c(-15,6))
## That's looking a lot better...
```

---

plot.manylm *Plot Diagnostics for a manylm or a manyglm Object*

Description

Four plots (selectable by `which`) are currently available: a plot of residuals against fitted values, a Normal Q-Q plot, a Scale-Location plot of $\sqrt{|residuals|}$ against fitted values, a plot of Cook's distances versus row labels. By default, all of them are provided.

The function is not yet available for manyglm object

Usage

```
## S3 method for class 'manylm'
plot(
  x, res.type="pearson", which=1:4, caption=c("Residuals vs Fitted", "Normal Q-Q",
  "Scale-Location", "Cook's distance"), overlay=TRUE,
  n.vars=Inf, var.subset=NULL, sub.caption=NULL, studentized= TRUE, ...)

## S3 method for class 'manyglm'
plot(
  x, res.type="pit.norm", which=1, caption=c("Residuals vs Fitted", "Normal Q-Q",
  "Scale-Location", "Cook's distance"), overlay=TRUE,
  n.vars=Inf, var.subset=NULL, sub.caption=NULL, ...)
```

Arguments

x	manylm object or manyglm object, typically the result of a call to manylm or manyglm.
res.type	type of residuals to plot. By default, res.type="pit-norm" uses Dunn-Smyth residuals (Dunn & Smyth 2996), related to the probability integral transform, for manyglm. These residuals are especially recommended for presence-absence data or discrete data.
which	if a subset of the plots is required, specify a subset of the numbers 1:4.
caption	captions to appear above the plots
overlay	logical, whether or not the different variables should be overlaid on a single plot.
n.vars	the number of variables to include in the plot.
var.subset	the variables to include in the plot.
sub.caption	common title—above figures if there are multiple; used as sub (s.title) otherwise. If NULL, as by default, a possible shortened version of deparse(x$call) is used.
...	other parameters to be passed through to plotting functions.
studentized	logical indicating whether studentized or standardized residuals should be used for plot 2 and 3.

Details

plot.manylm is used to check the linear model assumptions that are made when fitting a model via manylm. Similarly, plot.manyglm checks the generalised linear model assumptions made when using manyglm. As in Wang et al (2012), you should check the residual vs fits plot for no pattern (hence no suggestion of failure of key linearity and mean-variance assumptions). For manylm fits of small datasets, it is desirable that residuals on the normal Q-Q plot be close to a straight line, although in practice the most important thing is to make sure there are no big outliers and no suggestion of strong skew in the data.

The recommended res.type option for manyglm calls, "pit-norm", uses randomised quantile or "Dunn-Smyth" residuals (Dunn & Smyth 1996). Note that for discrete data, these residuals involve random number generation, and will not return identical results on replicate runs - so it is

recommended that you plot your data a few times to check if any pattern shows up consistently across replicate plots. The other main residual option is "pearson", Pearson residuals. Note that all res.type options are equivalent for manylm.

Some technical details on usage of this function:
`sub.caption` - by default the function call - is shown as a subtitle (under the x-axis title) on each plot when plots are on separate pages, or as a subtitle in the outer margin (if any) when there are multiple plots per page.

The 'Scale-Location' plot, also called 'Spread-Location' or 'S-L' plot, takes the square root of the absolute residuals in order to diminish skewness ($\sqrt{|E|}$ is much less skewed than $|E|$ for Gaussian zero-mean E).

If `studentized=FALSE` the 'S-L', the Q-Q, and the Residual-Leverage plot, use *standardized* residuals which have identical variance (under the hypothesis) otherwise *studentized* residuals are used.

Unlike other plotting functions `plot.manylm` and `plot.manyglm` respectively do not have a subset argument, the subset needs to be specified in the `manylm` or respectively `manyglm` function.

For all arguments that are formally located after the position of `...`, positional matching does not work.

For restrictions on `filename` see R's help on eps/pdf/jpeg. Note that `keep.window` will be ignored if `write.plot` is not `show`.

Author(s)

Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

References

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

Wang Y., Naumann U., Wright S.T. & Warton D.I. (2012). mvabund - an R package for model-based analysis of multivariate abundance data. Methods in Ecology and Evolution 3, 471-474.

See Also

`manylm`

Examples

```
require(graphics)

data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

## plot the diagnostics for the linear fit of the spider data
spidlm <- manylm(spiddat~X)
plot(spidlm,which=1:2,col.main="red",cex=3,overlay=FALSE)

plot(spidlm,which=1:4,col.main="red",cex=3,overlay=TRUE)
```

```
## plot the diagnostics for Poisson and negative binomial regression of the spider data
glmP.spid <- manyglm(spiddat~X, family="poisson")
plot(glmP.spid, which=1) #note the marked fan-shape on the plot

glmNB.spid <- manyglm(spiddat~X, family="negative.binomial")
plot(glmNB.spid, which=1) #no fan-shape
plot(glmNB.spid, which=1) #note the residuals change on re-plotting, but no consistent trend
```

`plot.mvabund` *Plot Multivariate Abundance Data and Formulae*

Description

Produces a range of plots for visualising multivariate abundance data and its relationship to environmental variables, including: dot-plots and boxplots for different levels of a factor stacked by response variable; comparative dot-plots and boxplots for different levels of a factor, stacked by response variable; scatterplots of abundances against a set of explanatory variables; scatterplots of pair-wise abundance data, e.g. from repeated measures. See details below.

Usage

```
## S3 method for class 'mvabund'
plot(x, y, type="p", overall.main="", n.vars=12,
  var.subset=NA, transformation="log", ...)

## S3 method for class 'mvformula'
plot(x,y=NA, type="p", var.subset=NA,
  n.vars= if(any(is.na(list(var.subset)))) 12 else length(var.subset),
  xvar.select=TRUE, xvar.subset = NA, n.xvars=NA, transformation="log", ...)
```

Arguments

`x`	for the `mvabund` method, x is a `mvabund` object. For the `mvformula` method, x is a `mvformula` object, a Model Formula to be used.
`y`	in `plot.mvabund` an optional second matrix with multivariate abundance data in `plot.mvformula` an optional matrix of the independent variables to explain x.
`type`	what type of plot should be drawn. Useful types are "p" for scatterplot, "bx" for boxplot and "n" for no plot. Other types, see `plot` are allowed, but usually don't give a meaningful output.
`overall.main`	a character to display as title for every window.
`var.subset`	a numeric vector of indices indicating which variables of the mvabund.object should be included on the plot.
`n.vars`	the number of variables to include in the plot.

xvar.select	whether only a subset of x variables should be plotted or all.
n.xvars	the number of the most relevant x variables that should be plotted, is not used if xvar.select = FALSE. If NA it will be set to at most 3.
xvar.subset	a subset of x variables that should be plotted, is not used if xvar.select = FALSE.
transformation	an optional transformation, if no formula is given, "no" = untransformed, "sqrt"=square root transformed, "log" (default)=log(Y/min+1) transformed, "sqrt4" =4th root transformed. Note that if plot.mvabund is called explicitly, and two data objects supplied, none of which is a mvabund object, then plot.mvformula will be called (See Details). The argument transformation is then NOT available.
...	arguments to be passed to or from other methods.

Details

The function plot.mvabund produces plots for the visualisation of multivariate abundance data and their relationships to environmental variables. The approach taken is to separately plot the relationship between each response variable and environmental variables, that is, to visualise the marginal distribution, as in Warton (2008). Three main types of plot that can be produced:

(1) Dot-plots or boxplots stacked along the y-axis by response variable. If a factor is given, comparative dot-plots/boxplots are produced, comparing abundances across each factor level. This type of plot is produced when one multivariate abundance dataset is given as an input argument, either on its own, or together with a factor, as in the examples using the solberg dataset below.

(2) Scatterplots of multivariate abundances against environmental variables, with separate plots for separate response variables. This type of plot is produced when one multivariate abundance dataset is given as an input argument together with an environmental variable or a set of environmental variables.

(3) Scatterplots of a paired sample of multivariate abundances. This type of plot is produced when two multivariate abundance datasets are given as input arguments, and their size and variable names match each other. It is up to the user to ensure that the rows match for these two datasets.

There are several methods for calling plot.mvabund:
(a) plot.mvabund("matrix", ...)
The multivariate abundances are stored in the data matrix. Including an optional second argument determines whether a plot of type (1) is produced (if no second argument or if it is a factor), or a plot of type (2) (if one or a set of environmental variables is given), or a plot of type (3) (if a second matching multivariate abundance dataset is given).
Instead of a matrix, mvabund objects can be used.
(b) plot("mvabund object", ...)
You can define mvabund objects using the function mvabund. Then the behaviour of the plot function is the same as plot.mvabund above.
(c) plot.mvformula("response"~"terms")
The first of these two objects must be the multivariate abundances, which can be either a matrix or a mvabund object. The terms determine the type of plot produced. The terms can be either a single vector or matrix or a number of vectors or matrices, separated by +. Compare formula for further details on the specification of the terms.
(d) plot("mvformula object")
You can define mvformula objects using the function mvformula. Note that the response cannot be a data frame object.

For plots of type (3) above, you must use method (a) or (b). Plot methods (c) and (d) require that both the response and explanatory variables are specified, i.e. formulas like '~x' or 'y~1' cannot be plotted.

See below examples to see how each of these methods is applied.

Multivariate abundance datasets typically have many variables, more than can be visualised in a single window, so by default plot.mvabund subsets abundance variables (and where appropriate, environmental variables). By default the 12 most abundant variables are plotted (determined on transformed variables if the response is transformed in the mvformula method), although this setting can be controlled via the argument `n.vars`, and the variables included in the subset to be plotted can be controlled via `var.subset`. It is possible for example to plot the abundance variables most significantly associated with an environmental variable, as in the Solberg example below.

To produce boxplots rather than dot-plots in type (1) plots, use the argument `type="bx"`.

For type (2) plots, if only one environmental variable is specified, plots for different abundance variables are arranged in a rectangular array (different abundance variables in different rows and columns). If however more than one environmental variable is specified, different columns correspond to different environmental variables (and different abundance variables in different rows). If more than 3 environmental variables are specified, the 3 will be selected that maximise average R^2 when `manylm` is applied (using the subset selection function `best.r.sq`). To switch off this subset selection, set `xvar.select=FALSE`, or choose your own subset of environmental variables using `xvar.subset`.

To control the appearance of points on dot-plots and scatterplots, usual arguments apply (see `par` for details). The plotting symbols `pch` and their `color` can be a vector, and if the plot function is called via a mvformula object, it can also be a list, where the list elements corresponds to the symbols / colors used in the plots for different independent variables.

If some of the formula terms are factor variables, these will be drawn in boxplots. Note, that the plots produced by `plot.mvformula` depend on whether the first independent variable is a factor or not. See the examples for the different possibilities of boxplots that can be produced.

If two objects are passed and only one of them is an `mvabund` object, the resulting plots will be the same as if a formula was supplied, having the `mvabund` object as response variable.
If both objects are not `mvabund` objects, it will be tried to guess which one of them is the response. The following logic applies: If `y` is not a `data.frame`, it will be assumed that `y` is the response. Note that `y` is the second object, if argument names are not supplied. If `y` is a `data.frame` and `x` is not a `data.frame`, it will be assumed that `x` is the response. If both objects are data frames an error will result, as the function is designed for `mvabund` objects!

The argument `shift` controls whether or not points are shifted on dotplots so that they do not overlap. This argument is ignored for boxplots and scatterplots (type (2) or type (3) graphs).

Warning

The argument `log`, that is available in lots of plotting functions can not be used for plotting `mvabund` or `mvformula` objects. Instead use `transformation` for the `mvabund` method and for the `mvformula` method include any transformations in the formula.

Author(s)

Ulrike Naumann, Yi Wang, Stephen Wright and David Warton <David.Warton@unsw.edu.au>.

References

Warton D.I. (2008). Raw data graphing: an informative but under-utilized tool for the analysis of multivariate abundances. *Austral Ecology* 33(3), 290-300.

See Also

boxplot.mvabund, meanvar.plot, plot.manylm, plot.manyglm.

Examples

```
require(graphics)

############################
## Some "type (1)" plots ##
############################

data(solberg)
solbdat <- solberg$abund
treatment<- solberg$x

## Plot a multivariate dataset (Species vs Abundance)
plot.mvabund(solbdat)

## Alternatively, the plot command could be used directly if spiddat is
## defined as an mvabund object:
solbmvabund <- mvabund(solbdat)
plot(solbmvabund)

## Draw an mvabund object in a boxplot, but using the 20 most abundant
## variables in the plot, using the square root transform, and adding
## coloured axes and title:
plot.mvabund(solbdat, n.vars=20, type="bx", transformation="sqrt",
fg="lightblue", main="Solberg abundances", col.main="red")

## Plot Species (split by treatment) vs Abundance
plot(solbmvabund,treatment)

## This can also be produced using
plot(solbmvabund~treatment)

## To use plot.mvabund to plot only the variables with P-values less than 0.1:
lm.solberg <- manylm(log(solbmvabund+1)~treatment)
anova.solb <- anova(lm.solberg, p.uni="unadjusted")
pj = anova.solb$uni.p

plot(solbmvabund~treatment, var.subset=pj<0.1)

## Or to plot only the 12 most significant variables, according to their
## univariate ANOVA P-values:
pj.sort = sort(pj, index.return=TRUE)
plot(solbmvabund~treatment, var.subset=pj.sort$ix[1:12])
```

```
############################
## Some "type (2)" plots ##
############################
#load and convert data
data(spider)
spiddat <- mvabund(spider$abund)
spidx <- mvabund(spider$x)

#create labels vectors
pch.vec <- as.numeric(spidx[,3]<2)
pch.vec[pch.vec==0] <- 3

#Scale the soil water variable
soilWater <- spidx[,1]

#Create the Table for the main titles of each plot
title <- c("\n\nAlopecosa accentuata", "\n\nAlopecosa cuneata",
           "\n\nAlopecosa fabrilis", "\n\nArctosa lutetiana",
           "\n\nArctosa perita", "\n\nAulonia albimana",
           "\n\nPardosa lugubris", "\n\nPardosa monticola",
           "\n\nPardosa nigriceps", "\n\nPardosa pullata",
           "\n\nTrochosa terricola", "\n\nZora spinimana")

#Plot Species Abundance vs Environmental variable
plot.mvformula(log(spiddat+1) ~ exp(soilWater), main=title,
xlab="% Soil Moist - Log Scale  ", ylab="Abundance [log scale]",
overall.main="Species Abundance vs %Soil Moisture", col=pch.vec,
fg="grey", pch=pch.vec, las=1, scale.lab="ss",t.lab="o", mfrow=c(4,3),log="x")

#Add a Margin
par(xpd=NA)
legend("topright",pch=c(1,3),col=c(1,3),legend = c("few twigs", "many twigs"),
cex = 1, inset=c(0,-0.19))


############################
## Some "type (3)" plots ##
############################

##Plot 1981 Abundance vs 1983 Abundance
data(tikus)
year <- tikus$x[,1]
tikusdat <- mvabund(tikus$abund)
site <- tikus$x[,2]

plot(tikusdat[year==81,], tikusdat[year==83,], col.main="blue",
xlab="1981 abundance", ylab="1983 abundance")
```

plotMvaFactor *Draw a Mvabund Object split into groups.*

Description

Draw the mvabund object x but split the data into groups according to the grouping variable y.

Usage

```
plotMvaFactor(x, y, type="p", main="Abundance", n.vars= min(12,NCOL(x)),
  transformation="log", legend=TRUE, ...)
```

Arguments

x	a mvabund object, a matrix with multivariate abundance data.
y	a factor or a data.frame with factors, non-factor columns in a data.frame are ignored.
type	what type of plot should be drawn, allowed types are "p" for scatterplot, "bx" for boxplot and "n" for no plot. Other types, as used in par are NOT allowed.
main	the title of the plot, see plot.
n.vars	the number of variables to include in the plot.
transformation	an optional transformation, "no" = untransformed, "sqrt"=square root transformed, "log" (default)=log(Y/min+1) transformed, "sqrt4" =4th root transformed.
legend	logical, whether a legend should be added to the plot.
...	arguments to be passed to or from other methods.

Details

For each variable in y that is a factor, a plot is drawn. When boxplots are drawn the colors, that can be supplied by col are used to display different factor levels. For scatterplots it is also possible to use the plotting symbols, specified by pch for that.

If the colors and for scatterplots the plotting symbols are not supplied, they will be automatically generated. However, the plotting symbols will only be automatically used in this way if there are up to seven different levels.

If colors or the plotting symbols are supplied, but the number of factor levels is bigger than the the number of different values, they will be replicated.

Sometimes the legends might be only partially visible, especially when the width of the graphics device is too small. To fix this, create a graphics device with a larger width (see help("device") for on available devices and their details) and then repeat the plotMvaFactor command.

Author(s)

Ulrike Naumann, Yi Wang, Stephen Wright and David Warton <David.Warton@unsw.edu.au>.

References

Warton, D. I. ( ) *Raw data graphing: an informative but under-utilised tool for the analysis of multivariate abundances*, , .

See Also

plot.mvabund.

Examples

```
require(graphics)

## Plot an Environment Factor vs Abundance plot
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

## Create a Environmental factor where TRUE=Sand, FALSE=No Sand)
X <- as.factor(X[,2]>0)
plotMvaFactor(x=spiddat, y=X)
```

---

predict.manyglm *Predict Method for MANYGLM Fits*

Description

Obtains predictions and optionally estimates standard errors of those predictions from a fitted manyglm object.

Usage

```
## S3 method for class 'manyglm'
predict(object, newdata, type = c("link", "response",
    "terms"), se.fit = FALSE, dispersion = NULL, terms = NULL,
    na.action = na.pass, ...)
```

Arguments

object	a fitted object of class inheriting from "manyglm".
newdata	optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
type	the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale. The value of this argument can be abbreviated.

se.fit	logical switch indicating if standard errors are required.
dispersion	the dispersion of the MANYGLM fit to be assumed in computing the standard errors. If omitted, that returned by summary applied to the object is used.
terms	with type="terms" by default all terms are returned. A character vector specifies which terms are to be returned
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	further arguments passed to or from other methods.

Details

predict.manyglm refits the model using glm before making predictions. In rare (usually pathological) cases this may lead to differences in predictions as compared to what would be expected if using the manyglm coefficients directly.

If newdata is omitted the predictions are based on the data used for the fit. In that case how cases with missing values in the original fit is determined by the na.action argument of that fit. If na.action = na.omit omitted cases will not appear in the residuals, whereas if na.action = na.exclude they will appear (in predictions and standard errors), with residual value NA. See also napredict.

Value

If se = FALSE, a matrix of predictions or an array of predictions and bounds. If se = TRUE, a list with components

fit	the predictions
se.fit	estimated standard errors
residual.scale	a scalar giving the square root of the dispersion used in computing the standard errors.

Author(s)

Ulrike Naumann, Yi Wang and David Warton <David.Warton@unsw.edu.au>.

See Also

manyglm.

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
Y <- spiddat[1:20, ]
X <- data.frame(spider$x[1:20, ])
glm.spid.poiss <- manyglm(Y~soil.dry+bare.sand, family="poisson", data=X)
glm.spid.poiss$data = X
newdata <- data.frame(spider$x[21:28,])
predict(glm.spid.poiss, newdata)
pred.w.plim <- predict(glm.spid.poiss, newdata, interval="prediction")
pred.w.clim <- predict(glm.spid.poiss, newdata, interval="confidence")
```

predict.manylm *Model Predictions for Multivariate Linear Models*

Description

predict.manylm is a function for predictions from the result of the model fitting function manylm.

Usage

```
## S3 method for class 'manylm'
predict(object, newdata=NULL, se.fit = FALSE,
    type = c("response", "terms"), terms = NULL, na.action = na.pass, ...)
```

Arguments

object	object of class inheriting from manylm.
newdata	an optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
se.fit	a switch indicating if standard errors are required.
type	type of prediction (response or model term), Possible values: "response", "terms".
terms	if type="terms", which terms (default is all terms).
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	further arguments passed to or from other methods.

Details

predict.manylm produces predicted values, obtained by evaluating the regression function in the frame newdata (which defaults to model.frame(object). If the logical se.fit is TRUE, standard errors of the predictions are calculated. If the numeric argument scale is set (with optional df), it is used as the residual standard deviation in the computation of the standard errors, otherwise this is extracted from the model fit. Setting intervals specifies computation of confidence or prediction (tolerance) intervals at the specified level, sometimes referred to as narrow vs. wide intervals.

If the fit is rank-deficient, some of the columns of the design matrix will have been dropped. Prediction from such a fit only makes sense if newdata is contained in the same subspace as the original data. That cannot be checked accurately, so a warning is issued.

If newdata is omitted the predictions are based on the data used for the fit. In that case how cases with missing values in the original fit is determined by the na.action argument of that fit. If na.action = na.omit omitted cases will not appear in the residuals, whereas if na.action = na.exclude they will appear (in predictions, standard errors or interval limits), with residual value NA. See also napredict.

The prediction intervals are for a single observation at each case in newdata (or by default, the data used for the fit) with error variance(s) pred.var. This can be a multiple of res.var, the estimated

value of σ^2: the default is to assume that future observations have the same error variance as those used for fitting. If `weights` is supplied, the inverse of this is used as a scale factor. For a weighted fit, if the prediction is for the original data frame, `weights` defaults to the weights used for the model fit, with a warning since it might not be the intended result. If the fit was weighted and newdata is given, the default is to assume constant prediction variance, with a warning.

Value

`predict.manylm` produces a matrix of predictions or if `interval` is set an array of predictions and bounds, where the first dimension has the names: `fit`, `lwr`, and `upr`. If `se.fit` is `TRUE`, a list with the following components is returned:

`fit`	vector or matrix as above
`se.fit`	a matrix with the standard errors of the predicted means
`residual.scale`	vector or matrix as a vector of residual standard deviations
`df`	numeric, the degrees of freedom for residual

Note

Variables are first looked for in `newdata` and then searched for in the usual way (which will include the environment of the formula used in the fit). A warning will be given if the variables found are not of the same length as those in `newdata` if it was supplied.

Offsets specified by `offset` in the fit by `lm` will not be included in predictions, whereas those specified by an offset term in the formula will be.

Notice that prediction variances and prediction intervals always refer to *future* observations, possibly corresponding to the same predictors as used for the fit. The variance of the *residuals* will be smaller.

Strictly speaking, the formula used for prediction limits assumes that the degrees of freedom for the fit are the same as those for the residual variance. This may not be the case if `res.var` is not obtained from the fit.

Author(s)

Ulrike Naumann, Yi Wang and David Warton <David.Warton@unsw.edu.au>.

See Also

`manylm`.

Examples

```
data(spider)
spiddat <- mvabund(spider$abund[1:20, ])
dat = data.frame(spider$x[1:20, ])
manylm.fit <- manylm(spiddat~soil.dry+bare.sand, data=dat)
predict(manylm.fit)
predict(manylm.fit, se.fit = TRUE)

new <- data.frame(spider$x[21:28, ])
```

```
predict(manylm.fit, new, se.fit = TRUE)
```

`predict.traitglm` *Predictions from fourth corner model fits*

Description

Obtains a prediction from a fitted fourth corner model object.

Usage

```
## S3 method for class 'traitglm'
predict(object, newR=NULL, newQ=NULL, newL=NULL, type="response", ...)
```

Arguments

`object`	a fitted object of class `traitglm`.
`newR`	A new data frame of environmental variables. If omitted, the original matrix of environmental variables is used.
`newQ`	A new data frame of traits for each response taxon. If omitted, the original matrix of traits is used.
`newL`	A new data frame of abundances (sites in rows, taxa in columns). This is only used if seeking predicted log-likelihoods. If omitted, the original abundances are used.
`type`	The type of prediction required. The default is predictions on the scale of the response variable, alternatives are `"logL"` for predictive log-likelihood, and "link" for linear predictors.
`...`	Further arguments passed to or from other methods.

Details

If `newR` and `newQ` are omitted, then as usual, predictions are based on the data used for the fit. Note that two types of predictions are possible in principle: predicting at new sites (by specifying a new set of environmental variables only, as `newR`) and predicting for new taxa (by specifying a new set of traits only, as `newQ`). Unfortunately, only predicting at new sites has been implemented at the moment! An issue with predicting to new taxa is that a main effect is included in the model for each taxon (by default), and the intercept would be unknown for a new species.

If predictive log-likelihoods are desired, a new data frame of abundances `newL` would need to be specified, whose rows correspond to those of `newR` and whose columns correspond to rows of `newQ`.

Value

A matrix of predictions, with sites in rows and taxa in columns.

Author(s)

David I. Warton <David.Warton@unsw.edu.au>

References

Brown AM, Warton DI, Andrew NR, Binns M, Cassis G and Gibb H (2014) The fourth corner solution - using species traits to better understand how species traits interact with their environment, Methods in Ecology and Evolution 5, 344-352.

See Also

traitglm

Examples

```
data(antTraits)

# fit a fourth corner model using negative binomial regression via manyglm:
ft=traitglm(antTraits$abund,antTraits$env,antTraits$traits,method="manyglm")
ft$fourth #print fourth corner terms

# predict to the first five sites
predict(ft, newR=antTraits$env[1:5,])
```

---

residuals.manyglm *Residuals for MANYGLM, MANYANY, GLM1PATH Fits*

Description

Obtains Dunn-Smyth residuals from a fitted manyglm, manyany or glm1path object.

Usage

```
## S3 method for class 'manyglm'
residuals(object, ...)
```

Arguments

object	a fitted object of class inheriting from "manyglm".
...	further arguments passed to or from other methods.

Details

`residuals.manyglm` computes Randomised Quantile or "Dunn-Smyth" residuals (Dunn & Smyth 1996) for a `manyglm` object. If the fitted model is correct then Dunn-Smyth residuals are standard normal in distribution.

Similar functions have been written to compute Dunn-Smyth residuals from `manyany` and `glm1path` objects.

Note that for discrete data, Dunn-Smyth residuals involve random number generation, and will not return identical results on replicate runs. Hence it is worth calling this function multiple times to get a sense for whether your interpretation of results holds up under replication.

Value

A matrix of Dunn-Smyth residuals.

Author(s)

David Warton <David.Warton@unsw.edu.au>.

References

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

See Also

`manyglm`, `manyany`, `glm1path`, `plot.manyglm`.

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

## obtain residuals for Poisson regression of the spider data, and doing a qqplot:
glmP.spid  <- manyglm(spiddat~X, family="poisson")
resP       <- residuals(glmP.spid)
qqnorm(resP)
qqline(resP,col="red")
#clear departure from normality.

## try again using negative binomial regression:
glmNB.spid <- manyglm(spiddat~X, family="negative.binomial")
resNB      <- residuals(glmNB.spid)
qqnorm(resNB)
qqline(resNB,col="red")
#that looks a lot more promising.

#note that you could construct a similar plot directly from the manyglm object using
plot(glmNB.spid, which=2)
```

ridgeParamEst *Estimation of the ridge parameter*

Description

Maximum likelihood estimation of the ridge parameter by cross-validation

Usage

```
ridgeParamEst(dat, X, weights = rep(1,times=nRows), refs,
tol=1.0e-010, only.ridge=FALSE,  doPlot=FALSE,
col="blue",type="l", ...)
```

Arguments

dat	the data matrix.
X	the design matrix.
weights	weights on the cases of the design matrix.
refs	a vector specifying validation group membership. Default is to construct refs using a method that is a function of the sample size N: if N<=20, leave-one-out is used refs=1:N, if N<=40, 10-fold Cross Validation is used where group membership is chosen randomly but with equal size groups, otherwise 5-fold CV with random group memberships.
tol	the sensitivity in calculations near zero.
only.ridge	logical, whether only the ridge Parameters should be passed back or additionally the Cross Validation penalised likelihood.
doPlot	logical, whether a plot of -2logL vs a candidate for the ridge parameter should be drawn.
col	color of Plot symbols.
type	type of Plot symbols.
...	further plot arguments.

Details

This function estimates the ridge parameter when applying ridge regularization to a sample correlation matrix of residuals. The ridge parameter is estimated to maximize the normal likelihood as estimated via cross validation (Warton 2008).

Value

A list with the following component:

ridgeParameter	the estimated ridge parameter

If only.ridge=FALSE the returned list additionally contains the element:

minLL	the minimum of the negative log-likelihood

.

Author(s)

David Warton <David.Warton@unsw.edu.au> and Ulrike Naumann.

References

Warton D.I. (2008). Penalized normal likelihood and ridge regularization of correlation and covariance matrices. *Journal of the American Statistical Association* 103, 340-349.

See Also

manylm

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

ridgeParamEst(dat = spiddat, X = model.matrix(spiddat~X))
```

---

shiftpoints *Calculate a shift for plotting overlapping points*

Description

Calculate a shift to add to overlapping points in plots for better visibility

Usage

```
shiftpoints(x, y, sh=( max(x)-min(x))/100, centered=TRUE, method=1, reg=6,
na.rm=TRUE)
```

Arguments

x	a data matrix or numeric vector for use in a plot.
y	a data matrix or numeric vector for use in a plot.
sh	the maximum total shift.
centered	logical, whether the shift is centered at 0, if FALSE the shift will be positive only.
method	numeric, can have the value 1 or 2, see Details.
reg	numeric, see Details.
na.rm	logical, indicating whether missing values should be removed.

Details

This function is similar to jitter but is defines for points in a two-dimensional plot. In contrast to jitter only the points with ties have a shift different from 0. The method to calculate the shift is therefore not based on random numbers.
If method=1 (default) the individual shift will be selected so that the shift range is sh, without regard of the number of overlapping points
method=2 means that for up to reg overlapping values a fixed shift of sh/reg is used

Value

Returns an array of shift values with the same dimension as x.

Author(s)

Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

See Also

plot.mvabund , plot.mvformula, jitter.

Examples

```
shiftpoints( x=c(1:5, 1:10), y=c(2:6, 1:10) )
```

---

solberg *Solberg Data*

Description

This dataset contains a list with abundance data of species and a factor variable.

Usage

```
data(solberg)
```

Format

A list containing the elements

abund a data frame containing 12 rows and has 53 variables, corresponding to the species. It has the following variables:
Paramesacanthion_sp., Halalaimus_sp., Viscosia_sp.,
Symplocostoma_sp., Bathylaimus_inermis, Bathylaimus_sp.,
Rhabdodemania_sp., Pandolaimus_latilaimus, Halanonchus_sp. ,
Trefusia_sp., Chromadora_sp., Dichromadora_sp.,
Neochromadora_sp., Prochromadorella_sp., Neotonchus_sp.,
Marylynnia_complexa, Paracanthonchus_sp., Cyatholaimidae_un .,
Choniolaimus_papillatus, Halichoanolaimus_dolichurus,

Richtersia_inaequalis, Dorylaimopsis_punctatus, Sabatieria_longicaudata, Sabatieria_punctata, Sabatieria_sp., Setosabieria_hilarula, Chromaspirina_sp., Molgolaimus_sp., Spirinia_parasitifera, Aponema_torosa, Microlaimus_sp.1, Microlaimus_sp.2, Camacolaimus_sp., Leptolaimus_elegans, Monhystera_sp., Amphimonhystera_sp., Daptonema_sp.1, Daptonema_sp.2, Daptonema_sp.3, Theristus_aff_acer, Xyalidae_un., Sphaerolaimus_macrocirculus, Sphaerolaimus_paradoxus, Desmolaimus_sp., Eleutherolaimus_sp., Eumorpholaimus_sp., Terschellingia_longicaudata, Paralinhomoeus_conicaudatus, Linhomieidae_un.A, Linhomieidae_un.B, Axonolaimus_sp., Odontophora_sp., Unidentified

x a factor with the levels control, high, low

Details

The abundance of each species was measured as the count of the number of organisms in the sample.

References

Gee J. M., Warwick R. M., Schaanning M., Berge J. A. and Ambrose W. G. Jr (1985) Effects of organic enrichment on meiofaunal abundance and community structure in sublittoral soft sediments. *Journal of Experimental Marine Biology and Ecology*. 91(3), 247-262.

Examples

```
data(solberg)
solbergdat <- mvabund( solberg$abund )
treatment <- solberg$x

## Create a formula for multivariate abundance data:
foo.sol <- mvformula( solbergdat ~ treatment )

## Fit a multivariate linear model:
lm.solberg <- manylm(foo.sol)
lm.solberg
```

---

spider *Spider data*

Description

data from spider2 directory, CANOCO FORTRAN package.

Usage

```
data(spider)
```

Format

A list containing the elements

abund A data frame with 28 observations of abundance of 12 hunting spider species

x A matrix of six (transformed) environmental variables at each of the 28 sites.

The data frame abund has the following variables

Alopacce (numeric) Abundance of the species Alopecosa accentuata

Alopcune (numeric) Abundance of the species Alopecosa cuneata

Alopfabr (numeric) Abundance of the species Alopecosa fabrilis

Arctlute (numeric) Abundance of the species Arctosa lutetiana

Arctperi (numeric) Abundance of the species Arctosa perita

Auloalbi (numeric) Abundance of the species Aulonia albimana

Pardlugu (numeric) Abundance of the species Pardosa lugubris

Pardmont (numeric) Abundance of the species Pardosa monticola

Pardnigr (numeric) Abundance of the species Pardosa nigriceps

Pardpull (numeric) Abundance of the species Pardosa pullata

Trocterr (numeric) Abundance of the species Trochosa terricola

Zoraspin (numeric) Abundance of the species Zora spinimana

The matrix x has the following variables

soil.dry (numeric) Soil dry mass

bare.sand (numeric) Cover bare sand

fallen.leaves (numeric) Cover fallen leaves / twigs

moss (numeric) Cover moss

herb.layer (numeric) Cover herb layer

reflection (numeric) Reflection of the soil surface with a cloudless sky

These variables have already been log(x+1)-transformed.

Details

The abundance of each species was measured as a count of the number of organisms in the sample.

Source

Data attributed to van der Aart & Smeenk-Enserink (1975), obtained from the spider2 directory, CANOCO FORTRAN package.

References

ter Braak, C. J. F. and Smilauer, P. (1998) CANOCO reference manual and user's guide to CANOCO for Windows: software for canonical community ordination (version 4). Microcomputer Power, New York, New York, USA.

van der Aart, P. J. M., and Smeenk-Enserink, N. (1975) Correlations between distributions of hunting spiders (Lycos- idae, Ctenidae) and environmental characteristics in a dune area. *Netherlands Journal of Zoology* **25**, 1-45.

Examples

```
require(graphics)

data(spider)
spiddat <- as.mvabund(spider$abund)

plot(spiddat)
```

`summary.manyglm` *Summarizing Multivariate Generalized Linear Model Fits for Abundance Data*

Description

`summary` method for class "manyglm".

Usage

```
## S3 method for class 'manyglm'
summary(object, resamp="pit.trap", test="wald",
        p.uni="none", nBoot=1000, cor.type=object$cor.type,
show.cor = FALSE, show.est=FALSE, show.residuals=FALSE,
symbolic.cor = FALSE, show.time=FALSE, show.warning=FALSE,...)
## S3 method for class 'summary.manyglm'
print(x, ...)
```

Arguments

`object`	an object of class `manyglm`, typically the result of a call to `manyglm`.
`resamp`	the method of resampling used. Can be one of "case", "perm.resid", "monte-carlo" or "pit.trap" (default). See Details.
`test`	the test to be used. If `cor.type="I"`, this can be one of "wald" for a Wald-Test or "score" for a Score-Test or "LR" for a Likelihood-Ratio-Test, otherwise only "wald" and "score" is allowed. The default value is "LR".

p.uni	whether to calculate univariate test statistics and their P-values, and if so, what type. This can be one of the following options. "none" = No univariate P-values (default) "unadjusted" = A test statistic and (ordinary unadjusted) P-value are reported for each response variable. "adjusted" = Univariate P-values are adjusted for multiple testing, using a step-down resampling procedure.
nBoot	the number of Bootstrap iterations, default is nBoot=999.
cor.type	structure imposed on the estimated correlation matrix under the fitted model. Can be "I"(default), "shrink", or "R". See Details.
show.cor, show.est, show.residuals	logical, if TRUE, the correlation matrix of the estimated parameters, or the estimated model parameters, or the residual summary is shown.
symbolic.cor	logical. If TRUE, the correlation is printed in a symbolic form (see symnum) rather than in numerical format.
show.time	Whether to display timing information for the resampling procedure: "none" shows none, "all" shows all timing information and "total" shows only the overall time taken for the tests.
show.warning	logical. Whether to display warnings in the operation procedure.
...	for summary.manyglm method, these are additional arguments including: rep.seed - logical. Whether to fix random seed in resampling data. Useful for simulation or diagnostic purposes. bootID - this matrix should be integer numbers where each row specifies bootstrap id's in each resampling run. When bootID is supplied, nBoot is set to the number of rows in bootID. Default is NULL. for print.summary.manyglm method, these are optional further arguments passed to or from other methods. See print.summary.glm for more details.
x	an object of class "summary.manyglm", usually, a result of a call to summary.manyglm.

Details

The summary.manyglm function returns a table summarising the statistical significance of each multivariate term specified in the fitted manyglm model (Warton 2011). For each model term, it returns a test statistic as determined by the argument test, and a P-value calculated by resampling rows of the data using a method determined by the argument resamp. Of the four possible resampling methods, three (case, residual permutation and parametric boostrap) are described in more detail in Davison and Hinkley (1997, chapter 6), but the default (PIT-trap) is a new method (in review) which bootstraps probability integral transform residuals, and which we have found to give the most reliable Type I error rates. All methods involve resampling under the alternative hypothesis. These methods ensure approximately valid inference even when the mean-variance relationship or the correlation between variables has been misspecified. Standardized pearson residuals (see manyglm are currently used in residual permutation, and where necessary, resampled response values are truncated so that they fall in the required range (e.g. counts cannot be negative). However, this can introduce bias, especially for family=binomial, so we advise extreme caution using perm.resid for presence/absence data. If resamp="none", p-values cannot be calculated, however the test statistics are returned.

If you have a specific hypothesis of primary interest that you want to test, then you should use the anova.manyglm function, which can resample rows of the data under the null hypothesis and so usually achieves a better approximation to the true significance level.

For information on the different types of data that can be modelled using manyglm, see manyglm. To check model assumptions, use plot.manyglm.

Multivariate test statistics are constructed using one of three methods: a log-likelihood ratio statistic test="LR", for example as in Warton et. al. (2012), or a Wald statistic test="wald" or a Score statistic test="score". "LR" has good properties, but is only available when cor.type="I".

The default Wald test statistic makes use of a generalised estimating equations (GEE) approach, estimating the covariance matrix of parameter estimates using a sandwich-type estimator that assumes the mean-variance relationship in the data is correctly specified and that there is an unknown but constant correlation across all observations. Such assumptions allow the test statistic to account for correlation between variables but to do so in a more efficient way than traditional GEE sandwich estimators (Warton 2008a). The common correlation matrix is estimated from standardized Pearson residuals, and the method specified by cor.type is used to adjust for high dimensionality.

The Wald statistic has problems for count data and presence-absence data when there are estimated means at zero (which usually means very large parameter estimates, check for this using coef). In such instances Wald statistics should not be used, Score or LR should do the job.

The summary.manyglm function is designed specifically for high-dimensional data (that, is when the number of variables p is not small compared to the number of observations N). In such instances a correlation matrix is computationally intensive to estimate and is numerically unstable, so by default the test statistic is calculated assuming independence of variables (cor.type="I"). Note however that the resampling scheme used ensures that the P-values are approximately correct even when the independence assumption is not satisfied. However if it is computationally feasible for your dataset, it is recommended that you use cor.type="shrink" to account for correlation between variables, or cor.type="R" when p is small. The cor.type="R" option uses the unstructured correlation matrix (only possible when N>p), such that the standard classical multivariate test statistics are obtained. Note however that such statistics are typically numerically unstable and have low power when p is not small compared to N.

The cor.type="shrink" option applies ridge regularisation (Warton (2008b)), shrinking the sample correlation matrix towards the identity, which improves its stability when p is not small compared to N. This provides a compromise between "R" and "I", allowing us to account for correlation between variables, while using a numerically stable test statistic that has good properties.

The shrinkage parameter is an attribute of the manyglm object. For a Wald test, the sample correlation matrix of the alternative model is used to calculate the test statistics. So object$shrink.param is used. For a Score test, the sample correlation matrix of the null model is used to calculate the test statistics. So shrink.param of the null model is used instead. If cor.type=="shrink" but object$shrink.param is not available, for example object$cor.type!="shrink", then the shrinkage parameter will be estimated by cross-validation using the multivariate normal likelihood function (see ridgeParamEst and (Warton 2008b)) in the summary test.

Rather than stopping after testing for multivariate effects, it is often of interest to find out which response variables express significant effects. Univariate statistics are required to answer this question, and these are reported if requested. Setting p.uni="unadjusted" returns resampling-based univariate P-values for all effects as well as the multivariate P-values, whereas p.uni="adjusted" returns adjusted P-values (that have been adjusted for multiple testing), calculated using a step-down resampling algorithm as in Westfall & Young (1993, Algorithm 2.8). This method provides

strong control of family-wise error rates, and makes use of resampling (using the method controlled by `resamp`) to ensure inferences take into account correlation between variables.

Value

summary.manyglm returns an object of class "summary.manyglm", a list with components

`call`	the component from `object`.
`terms`	the terms object used.
`family`	the component from `object`.
`deviance`	the component from `object`.
`aic`	Akaike's *An Information Criterion*, minus twice the maximized log-likelihood plus twice the number of coefficients (except for negative binomial and quasipoisson family, assuming that the dispersion is known).
`df.residual`	the component from `object`.
`null.deviance`	the component from `object`.
`df.null`	the component from `object`.
`devll`	minus twice the maximized log-likelihood
`iter`	the number of iterations that were used in `manyglm` for the estimation of the model parameters.
`p.uni`	the supplied argument.
`nBoot`	the supplied argument.
`resample`	the supplied argument.
`na.action`	the na.action used in the `manyglm` object, if applicable
`show.residuals`	the supplied argument.
`show.est`	the supplied argument.
`compositional`	logical. Whether a test for compositional effects was performed.
`test`	the supplied argument.
`cor.type`	the supplied argument.
`method`	the method used in `manyglm`. Either `"glm.fit"` or `"manyglm.fit"`
`theta.method`	the method used for the estimation of the nuisance parameter theta.
`manyglm.args`	a list of control parameters from `manyglm`.
`rankX`	the rank of the design matrix.
`covstat`	the supplied argument.
`deviance.resid`	the deviance residuals.
`est`	the estimated model coefficients
`s.err`	the Scaled Variance
`shrink.param`	the shrinkage parameter. Either the value of the argument with the same name or if this was not supplied the estimated shrinkage parameter.

n.bootsdone	the number of bootstrapping iterations that were done, i.e. had no error.
coefficients	the matrix of coefficients, standard errors, z-values and p-values. Aliased coefficients are omitted.
stat.iter	if the argument stat.iter is set to TRUE the test statistics in the resampling iterations.
statj.iter	if the argument stat.iter is set to TRUE the univariate test statistics in the resampling iterations.
aliased	named logical vector showing if the original coefficients are aliased.
dispersion	either the supplied argument or the inferred/estimated dispersion if the latter is NULL.
df	a 3-vector of the rank of the model and the number of residual degrees of freedom, plus number of non-aliased coefficients.
overall.n.bootsdone	the number of bootstrap iterations without errors that were done in the overall test
statistic	a table containing test statistics, p values and degrees of freedom for the overall test
overall.stat.iter	if the argument stat.iter is set to TRUE the test statistics of the overall tests in the resampling iterations.
overall.statj.iter	if the argument stat.iter is set to TRUE the univariate test statistics of the overall tests in the resampling iterations.
cov.unscaled	the unscaled (dispersion = 1) estimated covariance matrix of the estimated coefficients.
cov.scaled	ditto, scaled by dispersion.
correlation	(only if the argument show.cor = TRUE.) The estimated correlations of the estimated coefficients.
symbolic.cor	(only if show.cor = TRUE.) The value of the argument symbolic.cor.

Author(s)

Yi Wang, David Warton <David.Warton@unsw.edu.au> and Ulrike Naumann.

References

Warton D.I. (2011). Regularized sandwich estimators for analysis of high dimensional data using generalized estimating equations. *Biometrics*, 67(1), 116-123.

Warton D.I. (2008a). Penalized normal likelihood and ridge regularization of correlation and covariance matrices. *Journal of the American Statistical Association* 103, 340-349.

Warton D.I. (2008b). Which Wald statistic? Choosing a parameterisation of the Wald statistic to maximise power in *k*-sample generalised estimating equations. *Journal of Statistical Planning and Inference*, 138, 3269-3282.

Warton D. I., Wright S., and Wang, Y. (2012). Distance-based multivariate analyses confound location and dispersion effects. *Methods in Ecology and Evolution*, 3(1), 89-101.

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and their Application*, Cambridge University Press, Cambridge.

Westfall, P. H. and Young, S. S. (1993) *Resampling-based multiple testing.* John Wiley & Sons, New York.

Wu, C. F. J. (1986) Jackknife, Bootstrap and Other Resampling Methods in Regression Analysis. *The Annals of Statistics* 14:4, 1261-1295.

See Also

manyglm, anova.manyglm.

Examples

```
data(spider)
spiddat <- mvabund(spider$abund)
X <- spider$x

## Estimate the coefficients of a multivariate glm
glm.spid <- manyglm(spiddat[,1:3]~X, family="negative.binomial")

## Estimate the statistical significance of different multivariate terms in
## the model, using the default settings of LR test, and 100 PIT-trap resamples
summary(glm.spid, show.time=TRUE)

## Repeat with the parametric bootstrap and wald statistics
summary(glm.spid, resamp="monte.carlo", test="wald", nBoot=300)
```

---

summary.manylm *Summarizing Linear Model Fits for Multivariate Abundance Data*

Description

summary method for class "manylm" - computes a table summarising the statistical significance of different multivariate terms in a linear model fitted to high-dimensional data, such as multivariate abundance data in ecology.

Usage

```
## S3 method for class 'manylm'
summary(object, nBoot=1000, resamp="residual",
   test="F", cor.type=object$cor.type, shrink.param=NULL,
   p.uni="none", studentize=TRUE, R2="h", show.cor = FALSE,
   show.est=FALSE, show.residuals=FALSE, symbolic.cor=FALSE,
   tol=1.0e-6, ... )

## S3 method for class 'summary.manylm'
print(
```

```
x, digits = max(getOption("digits") - 3, 3),
signif.stars=getOption("show.signif.stars"),
dig.tst=max(1, min(5, digits - 1)),
eps.Pvalue=.Machine$double.eps, ... )
```

Arguments

object	an object of class "manylm", usually, a result of a call to manylm.
nBoot	the number of Bootstrap iterations, default is nBoot=1000.
resamp	the method of resampling used. Can be one of "case" (not yet available),"residual" (default), "perm.resid", "score" or "none". See Details.
test	the test to be used. Possible values are: "LR" = likelihood ratio statistic (default) and "F" = Lawley-Hotelling trace statistic. Note that if all variables are assumed independent (cor.shrink="I") then "LR" is equivalent to LR-IND and "F" is the sum-of-F statistics from Warton & Hudson (2004).
cor.type	structure imposed on the estimated correlation matrix under the fitted model. Can be "I"(default), "shrink", or "R". See Details.
shrink.param	shrinkage parameter to be used if cor.type="shrink". If not supplied, but needed, it will be estimated from the data by Cross Validation using the normal likelihood as in Warton (2008).
p.uni	whether to calculate univariate test statistics and their P-values, and if so, what type. "none" = no univariate P-values (default) "unadjusted" = a test statistic and (ordinary unadjusted) P-value is reported for each response variable. "adjusted" = Univariate P-values are adjusted for multiple testing, using a step-down resampling procedure.
studentize	logical, whether studentized residuals or residuals should beused for simulation in the resampling steps. This option is not used in case resampling.
R2	the type of R^2 (correlation coefficient) that should be shown, can be one of: "h" = Hooper's R^2 = tr(SST^(-1)SSR)/p "v" = vector R^2 = det(SSR)/det(SST) "n" = none
show.cor	logical, if TRUE, the correlation matrix of the estimated parameters is returned and printed.
show.est	logical. Whether to show the estimated model parameters.
show.residuals	logical. Whether to show residuals/a residual summary.
symbolic.cor	logical. If TRUE, print the correlations in a symbolic form rather than as numbers.
tol	the tolerance used in estimations.
x	an object of class "summary.manylm", usually, a result of a call to summary.manylm.
digits	the number of significant digits to use when printing.
signif.stars	logical. If TRUE, 'significance stars' are printed for each coefficient.
dig.tst	the number of digits to round the estimates of the model parameters.

eps.Pvalue	a numerical tolerance for the formatting of p values.
...	for summary.manyglm method, these are additional arguments including: rep.seed - logical. Whether to fix random seed in resampling data. Useful for simulation or diagnostic purposes. bootID - this matrix should be integer numbers where each row specifies bootstrap id's in each resampling run. When bootID is supplied, nBoot is set to the number of rows in bootID. Default is NULL. for print.summary.manyglm method, these are optional further arguments passed to or from other methods. See print.summary.glm for more details.

Details

The summary.manylm function returns a table summarising the statistical significance of each multivariate term specified in the fitted manylm model. For each model term, it returns a test statistic as determined by the argument test, and a P-value calculated by resampling rows of the data using a method determined by the argument resamp. The four possible resampling methods are residual-permutation (Anderson and Robinson (2001)), score resampling (Wu (1986)), case and residual resampling (Davison and Hinkley (1997, chapter 6)), and involve resampling under the alternative hypothesis. These methods ensure approximately valid inference even when the correlation between variables has been misspecified, and for case and score resampling, even when the equal variance assumption of linear models is invalid. By default, studentized residuals (r_i/sqrt(1-h_ii)) are used in residual and score resampling, although raw residuals could be used via the argument studentize=FALSE. If resamp="none", p-values cannot be calculated, however the test statistics are returned.

If you have a specific hypothesis of primary interest that you want to test, then you should use the anova.manylm function, which can resample rows of the data under the null hypothesis and so usually achieves a better approximation to the true significance level.

To check model assumptions, use plot.manylm.

The summary.manylm function is designed specifically for high-dimensional data (that, is when the number of variables p is not small compared to the number of observations N). In such instances a correlation matrix is computationally intensive to estimate and is numerically unstable, so by default the test statistic is calculated assuming independence of variables (cor.type="I"). Note however that the resampling scheme used ensures that the P-values are approximately correct even when the independence assumption is not satisfied. However if it is computationally feasible for your dataset, it is recommended that you use cor.type="shrink" to account for correlation between variables, or cor.type="R" when p is small. The cor.type="R" option uses the unstructured correlation matrix (only possible when N>p), such that the standard classical multivariate test statistics are obtained. Note however that such statistics are typically numerically unstable and have low power when p is not small compared to N. The cor.type="shrink" option applies ridge regularisation (Warton 2008), shrinking the sample correlation matrix towards the identity, which improves its stability when p is not small compared to N. This provides a compromise between "R" and "I", allowing us to account for correlation between variables, while using a numerically stable test statistic that has good properties. The shrinkage parameter by default is estimated by cross-validation using the multivariate normal likelihood function, although it can be specified via shrink.param as any value between 0 and 1 (0="I" and 1="R", values closer towards 0 indicate more shrinkage towards "I"). The validation groups are chosen by random assignment and so you may observe some slight variation in the estimated shrinkage parameter in repeat analyses. See ridgeParamEst for more details.

Rather than stopping after testing for multivariate effects, it is often of interest to find out which response variables express significant effects. Univariate statistics are required to answer this question, and these are reported if requested. Setting `p.uni="unadjusted"` returns resampling-based univariate P-values for all effects as well as the multivariate P-values, whereas `p.uni="adjusted"` returns adjusted P-values (that have been adjusted for multiple testing), calculated using a step-down resampling algorithm as in Westfall & Young (1993, Algorithm 2.8). This method provides strong control of family-wise error rates, and makes use of resampling (using the method controlled by `resample`) to ensure inferences take into account correlation between variables.

A multivariate R^2 value is returned in output, but there are many ways to define a multivariate R^2. The type of R^2 used is controlled by the `R2` argument. If `cor.shrink="I"` then all variables are assumed independent, a special case in which Hooper's R^2 returns the average of all univariate R^2 values, whereas the vector R^2 returns their product.

`print.summary.manylm` tries to be smart about formatting the coefficients, `genVar`, etc. and additionally gives 'significance stars' if `signif.stars` is `TRUE`.

Value

summary.manylm returns an object of class "summary.manyglm", a list with components

`call`	the component from `object`.
`terms`	the terms object used.
`show.residuals`	the supplied argument.
`show.est`	the supplied argument.
`p.uni`	the supplied argument.
`test`	the supplied argument.
`cor.type`	the supplied argument.
`resample`	the supplied argument.
`nBoot`	the supplied argument.
`rankX`	the rank of the design matrix
`residuals`	the model residuals
`genVar`	the estimated generalised variance
`est`	the estimated model coefficients
`shrink.param`	the shrinkage parameter. Either the value of the argument with the same name or if this was not supplied the estimated shrinkage parameter.
`aliased`	named logical vector showing if the original coefficients are aliased.
`df`	a 3-vector of the rank of the model and the number of residual degrees of freedom, plus number of non-aliased coefficients.

If the argument `test` is not `NULL` then the list also included the components

`coefficients`	a matrix containing the test statistics and the p-values.
`n.iter.sing`	the number of iterations that were skipped due to singularity of the design matrix caused by case resampling.

If furthermore the Design matrix is neither empty nor consists of the Intercept only, the following adddional components are included:

`r.squared`	the calculated correlation coefficient.
`R2`	a character that describes which type of correlation coefficient was calculated.
`statistic`	a matrix containing the results of the overall test.
`cov.unscaled`	the unscaled (`dispersion = 1`) estimated covariance matrix of the estimated coefficients.

If the argument `show.cor` is `TRUE` the following adddional components are returned:

`correlation`	the (p*q) by (p*q) correlation matrix, with p being the number of columns of the design matrix and q being the number of response variables. Note that this matrix can be very big.

Author(s)

Yi Wang, Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

References

Anderson, M.J. and J. Robinson (2001). Permutation tests for linear models. *Australian and New Zealand Journal of Statistics* 43, 75-88.

Davison, A. C. and Hinkley, D. V. (1997) *Bootstrap Methods and their Application.* Cambridge University Press, Cambridge.

Warton D.I. (2008). Penalized normal likelihood and ridge regularization of correlation and covariance matrices. *Journal of the American Statistical Association* 103, 340-349.

Warton D.I. and Hudson H.M. (2004). A MANOVA statistic is just as powerful as distance-based statistics, for multivariate abundances. *Ecology* 85(3), 858-874.

Westfall, P. H. and Young, S. S. (1993) *Resampling-based multiple testing.* John Wiley & Sons, New York.

Wu, C. F. J. (1986) Jackknife, Bootstrap and Other Resampling Methods in Regression Analysis. *The Annals of Statistics* 14:4, 1261-1295.

See Also

`manylm`, `anova.manylm`, `plot.manylm`

Examples

```
data(spider)
spiddat <- log(spider$abund+1)
spiddat <- mvabund(spiddat)
spidx <- spider$x

## Estimate the coefficients of a multivariate linear model:
fit <- manylm(spiddat~spidx)

## To summarise this multivariate fit, using score resampling to
```

```
## and F Test statistic to estimate significance:
summary(fit, resamp="score", test="F")

## Instead using residual permutation with 2000 iteration, using the sum of F
## statistics to estimate multivariate significance, but also reporting
## univariate statistics with adjusted P-values:
summary(fit, resamp="perm.resid", nBoot=2000, test="F", p.uni="adjusted")

## Obtain a summary of test statistics using residual resampling, accounting
## for correlation between variables but shrinking the correlation matrix to
## improve its stability and showing univariate p-values:
summary(fit, cor.type="shrink", p.uni="adjusted")
```

Tasmania *Tasmania Dataset*

Description

This dataset contains a list with community abundance data of species and two factor variables, namely treatment and block. See (Warwick et.al. (1990)) for more details.

Usage

```
data(Tasmania)
```

Format

A list containing the elements

abund A data frame with 16 observations of 56 Meiobenthos species exposed to a disturbance treatment in a spatially blocked design. Four blocks of four samples were collected such that each block comprised of two disturbed and undisturbed samples.

copepods A subset of abund of 12 Copepod species.

nematodes A subset of abund of 39 Nematode species.

treatment A two-level factor veraible.

block A four-level factor variable.

Details

The count data (number of each Meiobenthos species in each sample) were collected in a spatia lly blocked design. The labels are made to the four replicate cores within each block, with B labeling for the block ID and D labeling for the disturbed sample ID and U labeling for the undisturbed sample ID. The data frame abund contains 12 Copepod species, 39 Nematode species and 4 undetermined ones.

The 12 Copepod species are:
Ameira, Adopsyllus, Ectinosoma, Ectinosomat, Haloschizo,

Lepta.A, Lepta.B, Lepta.C, Mictyricola, Parevansula,
Quin, Rhizothrix

The 39 Nematode species are:
Actinonema, Axonolaimus, Bathylaimus,
Calyptronema, Chaetonema, Chromaspirina,
Comesoma, Daptonema, Desmodora.A,
Desmodora.B, Enoploides, Enoplus,
Epacanthion.A, Epacanthion.B, Eubostrichus,
Eurystomina, Hypodontolaimus.A, Hypodontolaimus.B,
Leptolaimus, Leptonemella, Mesacanthion,
Microlaimus, Monhystera, Nannoluimoides.A,
Nannolaimoides.B, Neochromadora.A, Neochromadora.B,
Odontophora, Oncholaimus, Qnvx,
Paracanthonchus, Polysigma, Praeacanthenchus,
Promonhystera, Pseudosteineria, Sabatieria,
Spilophorella, Symplocostoma, Viscosia

The data frame `copepod` stores the subset of 12 Copepod species, and the data frame `nematode` stores the subset of 39 Nematode species.

`treatment` indicates disturbed or undisturbed treatment for the 16 observations of each species in the Tasmania dataset.

`block` indicates the block ID for the 16 observations of each species in the Tasmania dataset.

References

Warwick, R. M., Clarke, K. R. and Gee, J. M. (1990). The effect of disturbance by soldier crabs Mictyris platycheles H. Milne Edwards on meiobenthic communiy structure. *J. Exp. Mar. Biol. Ecol.*, **135**, 19-33.

Examples

```
require(graphics)

data(Tasmania)
tasm.cop <- mvabund(Tasmania$copepods)
treatment <- Tasmania$treatment
block <- Tasmania$block

foo <- mvformula(tasm.cop~block*treatment)
plot(foo)
```

tikus *Tikus Island Dataset*

Description

This dataset contains a list with abundance data of species at different locations in the Tikus island and explanatory variables.

Usage

```
data(tikus)
```

Format

A list containing the elements

abund A data frame with 60 observations at different locations of abundances on 75 variables, the species. See Details.

x A data frame containing the id information for the Tikus island dataset. The data frame has 60 observations on 2 variables. See Details.

Details

The abundance of each species was measured as the length (in centimetres) of a 10 metre transect which intersected with the species.

`tikus` is a list containing the elements `abund` and `x`. The data frame `abund` contains 75 variables, the species:

Psammocora contigua, Psammocora digitata, Pocillopora damicornis, Pocillopora verrucosa, Stylopora pistillata, Acropora bruegemanni, Acropora robusta, Acropora grandis, Acropora intermedia, Acropora formosa, Acropora splendida, Acropera aspera, Acropora hyacinthus, Acropora palifera, Acropora cytherea, Acropora tenuis, Acropora pulchra, Acropora nasuta, Acropora humilis, Acropora diversa, Acropora digitifera, Acropora divaricata, Acropora subglabra, Acropora cerealis, Acropora valida, Acropora acuminata, Acropora elsevi, Acropora millepora, Montipora monasteriata, Montipora tuberculosa, Montipora hispida, Montipora digitata, Montipora foliosa, Montipora verrucosa, Fungia fungites, Fungia paumotensis, Fungia concina, Fungia scutaria, Halomitra limax, Pavona varians, Pavona venosa, Pavona cactus, Coeloseris mayeri, Galaxea fascicularis, Symphyllia radians, Lobophyllia corymbosa, Lobophyllia hemprichii, Porites cylindrica, Porites lichen, Porites lobata, Porites lutea, Porites nigrescens, Porites solida, Porites stephensoni, Goniopora lobata, Favia pallida, Favia speciosa, Favia stelligera, Favia rotumana, Favites abdita, Favites chinensis, Goniastrea rectiformis, Goniastrea pectinata, Goniastrea sp, Dulophyllia crispa, Platygyra daedalea, Platygyra sinensis, Hydnopora rigida, Leptastrea purpurea, Leptastrea pruinosa, Cyphastrea serailia, Millepora platyphylla, Millepora dichotoma, Millepora intrincata, Heliopora coerulea

`x` has the following variables:

time (factor) the year in which the measurement was taken.

rep (factor) the location id.

References

R.M. Warwick, K.R. Clarke and Suharsono (1990) A statistical analysis of coral community responses to the 19823 El Nino in the Thousand Islands, Indonesia, *Coral Reefs* **8**, 171179.

Examples

```
require(graphics)

data(tikus)

tikusdat <- as.mvabund(tikus$abund)
tikusid <- tikus$x
foo <- mvformula(tikusdat~tikusid[,1] + tikusid[,2])

plot(foo)
```

`traitglm` *Fits a fourth corner model for abundance as a function of environmental variables and species traits.*

Description

Fits a fourth corner model - a model to study how variation in environmental response across taxa can be explained by their traits. The function to use for fitting can be (pretty well) any predictive model, default is a generalised linear model, another good option is to add a LASSO penalty via `glm1path`. Can handle overdispersed counts via `family="negative.binomial"`, which is the default `family` argument.

Usage

```
traitglm(L, R, Q = NULL, family="negative.binomial", formula = NULL, method = "manyglm",
            composition = FALSE, col.intercepts = TRUE, ...)
```

Arguments

`L`	A data frame (or matrix) containing the abundances for each taxon (columns) across all sites (rows).
`R`	A data frame (or matrix) of environmental variables (columns) across all sites (rows).
`Q`	A data frame (or matrix) of traits (columns) across all taxa (rows). If not specified, a different environmental response will be specified for each taxon.
`family`	The family of the response variable, see `family`. Negative binomial with unknown overdispersion and a log-link can be specified as "negative.binomial", and is the default.

formula	A one-sided formula specifying exactly how to model abundance as a function of environmental and trait variables (as found in R and Q respectively). Default is to include all terms additively, with quadratics for quantitative terms, and all environment-by-trait interactions.
method	The function to use to fit the model. Default is manyglm, some other available options are glm1path, cv.glm1path for LASSO-penalised fits, but in principle any model-fitting function that accepts formula input and a family argument should work.
composition	logical. TRUE includes a row effect in the model, adjusting for different sampling intensities across different samples. This can be understood as a compositional term in the sense that all other terms then model relative abundance at a site. FALSE (default) does not include a row effect, hence the model is of absolute abundance.
col.intercepts	logical. TRUE (default) includes a column effect in the model, to adjust for different levels of abundance of different response (column) variables. FALSE removes this column effect.
...	Arguments passed to the function specified at method that will be used to fit the model.

Details

This function fits a fourth corner model, that is, a model to predict abundance across several taxa (stored in L) as a function of environmental variables (R) and traits (Q). The environment-trait interaction can be understood as the fourth corner, giving the set of coefficients that describe how environmental response across taxa varies as traits vary. A species effect is include in the model (i.e. a different intercept term for each species), so that traits are used to explain patterns in relative abundance across taxa not patterns in absolute abundance.

The actual function used to fit the model is determined by the user through the method argument. The default is to use manyglm to fit a GLM, although for predictive modelling, it might be better to use a LASSO penalty as in glm1path and cv.glm1path. In glm1path, the penalty used for BIC calculation is log(dim(L)[1]), i.e. log(number of sites).

The model is fitted by vectorising L then constructing a big matrix from repeated values of R, Q, their quadratic terms (if required) and interactions. Hence this function will hit memory issues if any of these matrices are large, and can slow down (especially if using cv.glm1path). If formula is left unspecified, the design matrix is constructed using all environmental variables and traits specified in R and Q, and quadratic terms for any of these variables that are quantitative, and all environment-trait interactions, after standardising these variables. Specifying a one-sided formula as a function of the variables in R and Q would instead give the user control over the precise model that is fitted, and drops the internal standardisations. The arguments composition and col.intercepts optionally add terms to the model for row and column total abundance, irrespective of whether a formula has been specified.

Note: when specifying a formula, if there are no penalties on coefficients (as for manyglm), then main effects for R can be excluded if including row effects (via composition=TRUE), and main effects for Q can be excluded if including column effects (via col.intercepts=TRUE), because those terms are redundant (trying to explain main effects for row/column when these main effects are already in the model). If using penalised likelihood (as in glm1path and cv.glm1path) or a

random effects model, by all means include main effects as well as row/column effects, and the penalties will sort out which terms to use.

If trait matrix `Q` is not specified, default behaviour will fit a different environmental response for each taxon (and the outcome will be very similar to `manyglm(L~R)`). This can be understood as a fourth corner model where species identities are used as the species traits (i.e. no attempt is made to explain differences across species).

These functions inherit default behaviour from their fitting functions. e.g. use `plot` for a Dunn-Smyth residual plot from a traits model fitted using `manyglm` or `glm1path`.

Value

Returns a `traitglm` object, a list that contains at least the following components:

... Exactly what is included in output depends on the fitting function - by default, a `manyglm` object is returned, so all usual `manyglm` output is included (coefficients, residuals, deviance, etc).

family A `family` object matching the final model.

fourth.corner A matrix of fourth corner coefficients. If `formula` has been manually entered, this will be a vector not a matrix.

R.des The reduced-size design matrix for environmental variables, including further arguments:

- **X** Data frame of (possibly standardised) environmental variables
- **X.squ** A data frame containing the leading term in a quadratic expression (where appropriate) for environmental variables
- **var.type** A vector with the same dimension as the number of columns of X, listing the type of ecah enviromental variable (`"quantitative"` or `"factor"`)
- **coefs** Coefficients used in transforming variables to orthogonality. These are used later to make predictions.

Q.des The reduced-size design matrix for traits, set up as for `R.des`.

spp.penalty For LASSO fits: a vector of the same length as the final design matrix, indicating which variables had a penalty imposed on them in model fitting.

L The data frame of abundances specified as input.

any.penalty Logical, is any penalty applied to parameters at all (not if using a `manyglm` fit).

scaling A list of coefficients describing the standaridsations of variables used in analyses. Stored for use later if making predictions.

call The original call `traitglm` call.

Author(s)

David I. Warton <David.Warton@unsw.edu.au>

References

Brown AM, Warton DI, Andrew NR, Binns M, Cassis G and Gibb H (2014) The fourth corner solution - using species traits to better understand how species traits interact with their environment, Methods in Ecology and Evolution 5, 344-352.

Warton DI, Shipley B & Hastie T (2015) CATS regression - a model-based approach to studying trait-based community assembly, Methods in Ecology and Evolution 6, 389-398.

See Also

glm1path, glm1, manyglm, family, residuals.manyglm, plot.manyany

Examples

```
data(antTraits)

ft=traitglm(antTraits$abund,antTraits$env,antTraits$traits,method="manyglm")
ft$fourth #print fourth corner terms

# for a pretty picture of fourth corner coefficients, uncomment the following lines:
# library(lattice)
# a        = max( abs(ft$fourth.corner) )
# colort   = colorRampPalette(c("blue","white","red"))
# plot.4th = levelplot(t(as.matrix(ft$fourth.corner)), xlab="Environmental Variables",
#   ylab="Species traits", col.regions=colort(100), at=seq(-a, a, length=100),
#   scales = list( x= list(rot = 45)))
# print(plot.4th)

plot(ft) # for a Dunn-smyth residual plot
qqnorm(residuals(ft)); abline(c(0,1),col="red") # for a normal quantile plot.

# predict to the first five sites
predict(ft,newR=antTraits$env[1:5,])

# refit using LASSO and less variables, including row effects and only two interaction terms:
ft1=traitglm(antTraits$abund,antTraits$env[,3:4],antTraits$traits[,c(1,3)],
     formula=~Shrub.cover:Femur.length+Shrub.cover:Pilosity,composition=TRUE,method="glm1path")
ft1$fourth #notice LASSO penalty has one interaction to zero
```

---

unabund *Remove the mvabund Class Attribute*

Description

Change an mvabund object to a non-mvabund object.

Usage

```
unabund(x)
```

Arguments

x	an mvabund object that should be transformed into a matrix.

Details

unabund doesn't convert x but only removes the mvabund class attribute.

Value

A matrix if x is an mvabund object otherwise x .

Author(s)

Ulrike Naumann and David Warton <David.Warton@unsw.edu.au>.

See Also

mvabund. as.mvabund. is.mvabund.

Examples

```
## Create an mvabund object:
abundances <- as.mvabund(matrix(1:20,5,4))

## Restore the original object:
mat <- unabund(x=abundances)
mat
```

Index

∗Topic **array**
manylm.fit, 47
∗Topic **classes**
mvabund, 50
unabund, 91
∗Topic **datasets**
antTraits, 19
solberg, 72
spider, 73
Tasmania, 85
tikus, 87
∗Topic **documentation**
mvabund-package, 2
∗Topic **dplot**
shiftpoints, 71
∗Topic **hplot**
boxplot.mvabund, 22
meanvar.plot, 48
plot.manyany, 53
plot.manylm, 54
plot.mvabund, 57
plotMvaFactor, 62
∗Topic **htest**
best.r.sq, 20
∗Topic **manip**
formulaUnimva, 29
unabund, 91
∗Topic **methods**
predict.manylm, 65
∗Topic **models**
anova.manyany, 5
anova.manyglm, 8
anova.manylm, 13
anova.traitglm, 17
best.r.sq, 20
cv.glm1path, 25
deviance.manylm, 27
extend.x.formula, 28
formulaUnimva, 29
glm1, 30
glm1path, 32
logLik.manylm, 35
manyany, 36
manyglm, 39
manylm, 44
mvformula, 52
predict.manyglm, 63
predict.traitglm, 67
residuals.manyglm, 68
ridgeParamEst, 70
summary.manyglm, 75
summary.manylm, 80
traitglm, 88
∗Topic **multivariate**
anova.manyany, 5
anova.manyglm, 8
anova.manylm, 13
anova.traitglm, 17
best.r.sq, 20
deviance.manylm, 27
formulaUnimva, 29
logLik.manylm, 35
manyany, 36
manyglm, 39
manylm, 44
manylm.fit, 47
mvabund, 50
mvformula, 52
plot.manyany, 53
plot.manylm, 54
predict.manyglm, 63
predict.manylm, 65
residuals.manyglm, 68
summary.manyglm, 75
summary.manylm, 80
∗Topic **regression**
anova.manyany, 5
anova.manyglm, 8

anova.manylm, 13
anova.traitglm, 17
best.r.sq, 20
cv.glm1path, 25
glm1, 30
glm1path, 32
manyany, 36
manyglm, 39
manylm, 44
manylm.fit, 47
plot.manyany, 53
plot.manylm, 54
predict.manyglm, 63
predict.traitglm, 67
residuals.manyglm, 68
summary.manyglm, 75
summary.manylm, 80
traitglm, 88
∗Topic **sysdata**
unabund, 91

anova.manyany, *3, 4*, 5, *37, 38*
anova.manyglm, *3, 4*, *6–8*, 8, *9*, *17, 18*, *41*, *43*, *77*, *80*
anova.manylm, *3, 4*, 13, *45, 46*, *84*
anova.traitglm, *3, 4*, 17
antTraits, *4*, 19
as.data.frame, *39*
as.mvabund, *92*
as.mvabund *(*mvabund*)*, 50
as.mvformula *(*mvformula*)*, 52

best.r.sq, 20, *29*
boxplot.mvabund, *3*, 22, *60*
boxplot.mvformula *(*boxplot.mvabund*)*, 22

coefficients, *41*
cv.glm1path, *3*, 25, *89*

deviance.manylm, 27

extend.x.formula, 28

family, *27*, *30–32*, *34*, *36*, *38, 39*, *41*, *88*, *91*
formula, *36, 37*, *39, 40*
formulaUnimva, *29*, 29

glm, *27*, *31*, *34*
glm1, *26, 27*, 30, *31*, *33, 34*, *91*
glm1path, *3*, *25–27*, *31*, 32, *53, 54*, *69*, *89–91*

is.mvabund, *92*
is.mvabund *(*mvabund*)*, 50
is.mvformula *(*mvformula*)*, 52

jitter, *72*

lm, *37*, *40*, *66*
logLik.manylm, 35

manyany, *2–4*, *6*, *8*, 36, *37*, *41*, *53, 54*, *69*
manyglm, *3, 4*, *9–12*, *17, 18*, *33*, *37*, 39, *53, 54*, *64*, *69*, *75–78*, *80*, *89–91*
manylm, *3, 4*, *13*, *16*, *28*, 44, *47, 48*, *52*, *55, 56*, *66*, *71*, *81*, *84*
manylm.fit, *45*, 47
manylm.wfit, *45*
manylm.wfit *(*manylm.fit*)*, 47
meanvar.plot, *3, 4*, 48, *60*
meanvar.plot,data.frame-method *(*meanvar.plot*)*, 48
meanvar.plot,formula-method *(*meanvar.plot*)*, 48
meanvar.plot,matrix-method *(*meanvar.plot*)*, 48
meanvar.plot,mvabund-method *(*meanvar.plot*)*, 48
meanvar.plot,mvformula-method *(*meanvar.plot*)*, 48
meanvar.plot.mvabund *(*meanvar.plot*)*, 48
meanvar.plot.mvformula *(*meanvar.plot*)*, 48
mvabund, *4*, *29*, 50, *52*, *92*
mvabund-class *(*mvabund*)*, 50
mvabund-package, 2
mvformula, *4*, *29*, *51*, 52
mvformula-class *(*mvformula*)*, 52

na.exclude, *37*, *40*
na.fail, *40*
na.omit, *40*
napredict, *64, 65*

options, *40*

plot.glm1path *(*plot.manyany*)*, 53
plot.manyany, *7*, *34*, *38*, 53, *91*
plot.manyglm, *3*, *10*, *43*, *60*, *69*, *77*
plot.manyglm *(*plot.manylm*)*, 54
plot.manylm, *3, 4*, *16*, *46*, 54, *60*, *84*
plot.mvabund, *3, 4*, *24*, *49*, *51*, 57, *63*, *72*

plot.mvformula, *29*, *49*, *72*
plot.mvformula *(*plot.mvabund*)*, 57
plotMvaFactor, 62
predict.manyglm, 63
predict.manylm, 65
predict.traitglm, 67
print.anova.manyany *(*anova.manyany*)*, 5
print.anova.manyglm *(*anova.manyglm*)*, 8
print.anova.manylm *(*anova.manylm*)*, 13
print.manyany *(*manyany*)*, 36
print.summary.glm, *9*, *76*, *82*
print.summary.manyglm
*(*summary.manyglm*)*, 75
print.summary.manylm *(*summary.manylm*)*, 80

residuals.glm1path *(*residuals.manyglm*)*, 68
residuals.manyany, *38*
residuals.manyany *(*residuals.manyglm*)*, 68
residuals.manyglm, *34*, *43*, *53*, 68, *91*
ridgeParamEst, *11*, *15*, *41*, 70, *77*, *82*

sample, *6*
shiftpoints, 71
solberg, *4*, 72
spider, *4*, 73
summary.manyglm, *3*, *12*, *41*, *43*, 75
summary.manylm, *3, 4*, *16*, *45, 46*, 80
symnum, *76*

Tasmania, *4*, 85
terms, *42*
tikus, *4*, 87
traitglm, *3, 4*, *17, 18*, *68*, 88

unabund, *51*, *91*, 91