

# Package ‘mvabund’

January 27, 2015

**Title** Statistical Methods for Analysing Multivariate Abundance Data

**Version** 3.9.3

**Date** 02-11-2014

**Author** Yi Wang, Ulrike Naumann, Stephen Wright and David Warton

**Maintainer** Yi Wang <alice.yi.wang@gmail.com>

**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 (>= 3.0.2), Rcpp (>= 0.11)

**LinkingTo** Rcpp (>= 0.11)

**Imports** MASS, methods, stats, tweedie, statmod

**SystemRequirements** GNU GSL (>= 1.10)

**License** LGPL (>= 2.1)

**Repository** CRAN

**NeedsCompilation** yes

**Date/Publication** 2014-11-03 00:39:18

## R topics documented:

mvabund-package . . . . .	2
anova.manyany . . . . .	5
anova.manyglm . . . . .	8
anova.manylm . . . . .	12
best.r.sq . . . . .	16
boxplot.mvabund . . . . .	17
deviance.manylm . . . . .	20
extend.x.formula . . . . .	21
formulaUnimva . . . . .	22

logLik.manylm . . . . .	23
manyany . . . . .	24
manyglm . . . . .	26
manylm . . . . .	31
manylm.fit . . . . .	34
meanvar.plot . . . . .	35
mvabund . . . . .	37
mvformula . . . . .	39
plot.manyany . . . . .	40
plot.manylm . . . . .	41
plot.mvabund . . . . .	44
plotMvaFactor . . . . .	48
predict.manyglm . . . . .	50
predict.manylm . . . . .	52
residuals.manyglm . . . . .	54
ridgeParamEst . . . . .	55
shiftpoints . . . . .	57
solberg . . . . .	58
spider . . . . .	59
summary.manyglm . . . . .	61
summary.manylm . . . . .	66
Tasmania . . . . .	71
tikus . . . . .	72
unabund . . . . .	74
<b>Index</b>	<b>75</b>

---

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)).

This includes 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, and diagnostic plots to check the appropriateness of a fitted model (Wang et. al (2012)).

Future versions of this function will extend it to regression analysis using additive functions, and model-based cluster analysis.

## 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  
`plotMvaFactor` draw a range of plots for Multivariate Abundance Data where the data is split into groups according to a grouping variable

#### **For estimating and displaying Linear Models:**

`manylm` Fitting Linear Models for Multivariate Abundance Data  
`summary.manylms` summarize Multivariate Linear Model Fits for Abundance Data  
`anova.manylms` obtain ANOVA for Multivariate Linear Model Fits for Abundance Data  
`plot.manylms` plot diagnostics for a `manylms` 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

#### **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 data with a 3-level treatment factor.  
`spider` hunting spiders data from different sites.  
`tikus` solberg nematode data from Tikus island.

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**

- Lawless, J. F. (1987) *Negative binomial and mixed Poisson regression*, Canadian Journal of Statistics 15, 209-225.
- Liang, K.-Y. and Zeger, S. L (1986) *Longitudinal Data Analysis Using Generalized Linear Models*, Biometrika 73, No. 1, 13-22.
- Warton D.I. (2008a). Raw data graphing: an informative but under-utilized tool for the analysis of multivariate abundances. *Austral Ecology* 33(3), 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. & Hudson H.M. (2004). A MANOVA statistic is just as powerful as distance-based statistics, for multivariate abundances. *Ecology* 85(3), 858-874.

Warton, D. I. and Guttorp, P. (in press April 2010). Compositional analysis of overdispersed counts using generalized estimating equations. *Environmental and Ecological Statistics*.

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

[mvabund](#), [mvformula](#), [plot.mvabund](#), [boxplot.mvabund](#), [meanvar.plot](#), [plotMvaFactor](#), [many1m](#), [summary.many1m](#), [anova.many1m](#), [plot.many1m](#),

### 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 (2008):
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	<i>Analysis of Deviance for Many Univariate Models Fitted to Multivariate Abundance Data</i>
---------------	--

---

### 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, replace=TRUE)
## S3 method for class 'anova.manyany'
print(object)

```

### Arguments

object	of class manyany under the null hypothesis, typically the result of a call to <a href="#">manyany</a> .
...	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).
replace	whether to sample with or without replacement, as in the <a href="#">sample</a> function. = FALSE for PIT-permutation, = TRUE for PIT-trap.

## 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

<code>stat</code>	the observed value of the test statistic.
<code>p</code>	the P-value as estimated from <code>nBoot</code> 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
##avoid errors and cut computation time. Worth rerunning 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 several seconds to run even for just 9 resamples
## This should be rerun for nBoot=999, which would take closer to an hour...
## Results though are worth the effort, they suggest no evidence that composition changed!
## (Although there was a very strong signal of a change in total abundance)
## (And the Tweedie model has some issues for this dataset, as seen in a plot of the fit)
```

---

anova.manyglm	<i>Analysis of Deviance for Multivariate Generalized Linear Model Fits for Abundance Data</i>
---------------	---

---

## 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=1000, cor.type=object$cor.type,
       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 <a href="#">manyglm</a> .
...	for the anova.manyglm method, these are optional further objects of class manyglm, which are usually a result of a call to <a href="#">manyglm</a> for the print.anova.manyglm method these are optional further arguments passed to or from other methods. See <a href="#">print.summary.glm</a> 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.



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 "summary.manyglm", usually, a result of a call to <a href="#">summary.manyglm</a> .

## 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 zero parameters, so is not recommended for multi-sample tests, where such situations are common.

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

<code>family</code>	the family component from object.
<code>p.uni</code>	the <code>p.uni</code> argument supplied.
<code>test</code>	the test argument supplied.
<code>cor.type</code>	the <code>cor.type</code> argument supplied.
<code>resamp</code>	the <code>resamp</code> argument supplied.
<code>nBoot</code>	the <code>nBoot</code> argument supplied.
<code>shrink.parameter</code>	a list of shrink parameters from all <code>manyglm</code> 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.

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$scopepods)
treatment <- Tasmania$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 the default settings of parametric resampling and deviance as test statistics:
anova(tasm.cop.nb, nBoot=200, test="wald")
```

---

anova.manylm

*ANOVA for Linear Model Fits for Multivariate Abundance Data*


---

## Description

anova method for class "manyglm" - 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 'manyglm'
anova(object, ..., resamp="perm.resid", test="F", p.uni="none",
       nBoot=1000, 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.manyglm'
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 manyglm, usually, a result of a call to <a href="#">manyglm</a> .
...	for the anova.manyglm method, these are optional further objects of class manyglm, which are usually a result of a call to <a href="#">manyglm</a> . for the print.anova.manyglm method these are optional further arguments passed to or from other methods.
nBoot	the number of iterations in resampling.
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:

<code>p.uni</code>	the supplied argument.
<code>test</code>	the supplied argument.
<code>cor.type</code>	the supplied argument.
<code>resample</code>	the supplied argument.
<code>nBoot</code>	the supplied argument.
<code>calc.rss</code>	the supplied argument.
<code>table</code>	the data frame containing the anova table.
<code>shrink.param</code>	the supplied argument.
<code>n.bootsdone</code>	the number of bootstrapping iterations that were done, i.e. had no error.
<code>n.iter.sing</code>	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
## 5000 resamples, and also reporting univariate statistics with
## adjusted P-values:
anova(fit, fit0, fit1, fit2, nBoot=5000, test="F", p.uni="adjust")

```

---

best.r.sq

*Use R<sup>2</sup> 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<sup>2</sup>. 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 <sup>2</sup> that should be found.
R2	the type of R <sup>2</sup> (correlation coefficient) that should be shown, possible values are: " h " = Hooper's R <sup>2</sup> = $\text{tr}(\text{SST}^{-1}\text{SSR})/p$ " v " = vector R <sup>2</sup> = $\text{det}(\text{SSR})/\text{det}(\text{SST})$ " 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 )
```

### 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 <code>mvabund</code> method <code>x</code> 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 <code>x</code> 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 <code>pars</code> (and override those in <code>pars</code> ). For the <code>mvformula</code> method, a formula, such as <code>y ~ grp</code> , where <code>y</code> is a numeric <code>mvabund</code> object of data values to be split into groups according to the grouping variable <code>grp</code> (a factor).
y	for the <code>mvabund</code> method <code>y</code> can be an additional <code>mvabund</code> object, if <code>x</code> isn't a list.
range	this determines how far the plot whiskers extend out from the box. If <code>range</code> is positive, the whiskers extend to the most extreme data point which is no more than <code>range</code> 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 <code>mvabund</code> method: group labels which will be printed under each boxplot.
at	only available for the <code>mvabund</code> method: numeric vector giving the locations where the boxplots should be drawn; defaults to <code>1:n</code> where <code>n</code> 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 <code>mvabund</code> object should be included on the plot.
transformation	an optional transformation, (ONLY) for the <code>mvabund</code> method. Note, that for the <code>mvabund</code> method <code>transformation</code> must be used instead of <code>log</code> . Available values are: "no" = untransformed, "sqrt"=square root transformed, "log" (default)= $\log(Y/\min+1)$ transformed, "sqrt4" =4th root transformed.
...	for the <code>mvformula</code> method, named arguments to be passed to the <code>plot.mvformula</code> method. Some arguments that are available for the <code>mvabund</code> method, are not available in <code>plot.mvformula</code> and can therefore not available in the <code>mvformula</code> method.

For the mvabund method, unnamed 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 available 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)
```

---

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)
```

---

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") (degrees of freedom) 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,
var.power=NA, ...)

```

**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 " <b>formula</b> " (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 <b>REQUIRED</b> 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 <b>family</b> 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. <b>FALSE</b> (default) fits a separate model to each species. <b>TRUE</b> fits a single model to all variables, including site as a row effect, such that all other terms model relative abundance (compositional effects).
var.power	the power parameter, if using the tweedie distribution.
...	further arguments passed to the fitting function.



## 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'). 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 using 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:

<code>logL</code>	a vector of log-likelihood terms for each response variable in the fitted model.
<code>fitted.values</code>	the matrix of fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
<code>residuals</code>	the matrix of probability integral transform (PIT) residuals. If the fitted model is a good fit, these will be approximately standard uniformly distributed.
<code>linear.predictor</code>	the linear fit on link scale.
<code>family</code>	a vector of <code>family</code> arguments, one for each response variable.
<code>call</code>	the matched call.
<code>model</code>	the <code>model.frame</code> from the model for the last response variable.
<code>terms</code>	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 poisson:
spidPois <- manyany("glm", abund, data=X, abund~X, family=poisson())

logLik(spidPois) # a number of generic functions are applicable to manyany objects
```

---

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 " <b>formula</b> " (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 <b>Details</b> .   |
| family  | a description of the error distribution function to be used in the model. At the moment, this must be a character string naming one of the following members of the exponential family: 'gaussian', 'poisson', 'binomial', 'negative.binomial'. Further improvement will include options of specifying the link function and calling to a family function defined in <a href="#">family</a> . |
| 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 <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>glm</code> 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 <code>na.action</code> setting of <code>options</code> , and is <code>na.fail</code> if that is unset. The ‘factory-fresh’ default is <code>na.omit</code> . Another possible value is <code>NULL</code> , no action. Value <code>na.exclude</code> can be useful.
theta.method	the method used for the estimation of the overdispersion 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 <code>manyglm</code> , and will be used as the default value for <code>cor.type</code> in subsequent functions for inference.
shrink.param	shrinkage parameter to be used if <code>cor.type="shrink"</code> . 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 <code>manyglm</code> object, and will be used in subsequent functions for inference.
tol	the tolerance used in estimations.
maxiter	maximum allowed iterations in the weighted least square estimation of beta. The default value is 100.
maxiter2	maximum allowed iterations in the internal ML estimations of negative binomial regression. The default value is 5.
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 <code>_a priori_</code> known component to be included in the linear predictor during fitting. This should be 'NULL' or a numeric vector of length equal to <code>NROW</code> (i.e. number of observations) or a matrix of <code>NROW</code> times <code>p</code> (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 this compare the details section of [lm](#) and [formula](#). A formula has an implied intercept term. To remove this use either  $y \sim x - 1$  or  $y \sim 0 + x$ . See [formula](#) for more details of allowed formulae.

Generalised linear models are designed for non-normal data where a specific alternative distribution offers a reasonable model for data, as specified using the argument family (Warton (2011)). Count data can be analysed using family="poisson" if the data is 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 a good choice of model for counts (Warton (2005)). For binary (presence/absence) data, family="binomial" should be used.

The negative binomial function requires knowledge of the overdispersion parameter (theta) in order to fit a generalised linear model. However, it is usual not to have knowledge of the value of theta. So we estimate

a separate theta for each variable from the data. The method of estimating theta is controlled by theta.method for negative binomial distributions.

family not only chooses the type of distribution to fit to data, but also controls the link function - the function of the mean that is fitted with a linear predictor. The default is the canonical link function, except for negative binomial, which uses the log-link.

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", "manyglm" and "mgglm".

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.

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.

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

linear.predictors	the linear fit on link scale.
residuals	the matrix of <i>working</i> residuals, that is the Pearson residuals standardized by the leverage adjustment $h$ obtained from the diagonal elements of the hat matrix $H$ .
sqrt.1_Hii	the matrix of scale terms used to standardize the Pearson residuals.
variance.estimator	the variance estimator for the corresponding family function.
sqrt.weight	the matrix of square root of <i>working</i> weights, estimated for the corresponding family function.
theta	the estimated nuisance parameters accounting for overdispersion
phi	the inverse of theta
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.
AIC	<i>Akaike's An Information Criterion</i> , minus twice the maximized log-likelihood plus twice the number of coefficients.
AICsum	the sum of the AIC's over all variables.
iter	the number of iterations of IWLS used.
tol	the tolerance used in estimations.
family	the <code>family</code> argument supplied.
theta.method	the <code>theta.method</code> argument supplied.
cor.type	the <code>cor.type</code> argument supplied.
shrink.param	the shrink parameter to be used in subsequent inference.
call	the matched call.
terms	the <code>terms</code> object used.
formula	the formula supplied.
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.
model	if the argument <code>model</code> is TRUE, this is the <code>model.frame</code> .
y	if the argument <code>y</code> is TRUE, this is the response variables used.
x	if the argument <code>x</code> is TRUE, this is the design matrix used.
qr	if the argument <code>qr</code> is TRUE, this is the QR decomposition of the design matrix.

Objects of class "glm" are normally of class `c("glm", "lm")`, that is inherit from class "lm", and well-designed methods for class "lm" will be applied to the weighted linear model at the final iteration of IWLS. However, care is needed, as extractor functions for class "glm" such as `residuals` and `weights` do **not** just pick out the component of the fit with the same name.

#### 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](#),

## 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.
```

---

 many1m
 

---



---

*Fitting Linear Models for Multivariate Abundance Data*


---

### Description

many1m 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.many1m` for assumption checking, and `anova.many1m` or `summary.many1m` for significance testing.

### Usage

```
many1m(
  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 <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>many1m</code> 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 <code>NULL</code> or a numeric vector. If non-null, weighted least squares is used with weights <code>weights</code> (that is, minimizing $\sum(\text{weights} * e^2)$ ); 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 <code>na.action</code> setting of <code>options</code> , and is <code>na.fail</code> if that is unset. The 'factory-fresh' default is <code>na.omit</code> . Another possible value is <code>NULL</code> , no action. Value <code>na.exclude</code> can be useful.
method	the method to be used; for fitting, currently only <code>method = "qr"</code> is supported; <code>method = "model.frame"</code> returns the model frame (the same as with <code>model = TRUE</code> , see below).
model, x, y, qr	logicals. If <code>TRUE</code> the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
singular.ok	logical. If <code>FALSE</code> (the default in S but not in R) a singular fit is an error.

<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>offset</code>	this can be used to specify an <i>a priori</i> known component to be included in the linear predictor during fitting. This should be <code>NULL</code> or a numeric vector of length either one or equal to the number of cases. One or more <code>offset</code> terms can be included in the formula instead or as well, and if both are specified their sum is used. See <code>model.offset</code> .
<code>test</code>	choice of test statistic. Can be one of "LR" (default) = likelihood ratio statistic "F" = Lawley-Hotelling trace statistic <code>NULL</code> = no test This parameter is merely stored in <code>many1m</code> , and will be used as the default value of <code>test</code> in subsequent functions for inference.
<code>cor.type</code>	structure imposed on the estimated correlation matrix under the fitted model. Can be "I"(default), "shrink", or "R". See <code>anova.many1m</code> for details of its usage. This parameter will be used as the default value of <code>cor.type</code> in subsequent functions for inference.
<code>shrink.param</code>	shrinkage parameter to be used if <code>cor.type="shrink"</code> . This parameter will be used as the default value of <code>shrink.param</code> in subsequent functions for inference.
<code>tol</code>	the tolerance used in estimations.
<code>...</code>	additional arguments to be passed to the low level regression fitting functions (see below).

## Details

Models for `many1m` 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  $y \sim x - 1$  or  $y \sim 0 + x$ . See `formula` for more details of allowed formulae.

`many1m` calls the lower level function `many1m.fit` or `many1m.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.many1m` or `anova.many1m`.

## Value

`many1m` returns an object of `c("many1m", "mlm", "lm")` for multivariate formula response and of class `c("lm")` for univariate response.

A `many1m` object is a list containing at least the following components:

<code>coefficients</code>	a named matrix of coefficients
<code>residuals</code>	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.manyl`.

### Author(s)

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

### See Also

[anova.manyl](#), [summary.manyl](#), [plot.manyl](#)

### 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

 workhorse 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

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

### Value

	a list with components
<code>coefficients</code>	$p$ vector
<code>residuals</code>	$n$ vector or matrix
<code>fitted.values</code>	$n$ vector or matrix
<code>weights</code>	$n$ vector — <i>only</i> for the <code>*wfit*</code> functions.
<code>rank</code>	integer, giving the rank
<code>qr</code>	(not null fits) the QR decomposition.
<code>df.residual</code>	degrees of freedom of residuals
<code>hat.X</code>	the hat matrix.
<code>txX</code>	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**

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

<code>check.rows</code>	if TRUE then the rows are checked for consistency of length and names.
<code>check.names</code>	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 <code>make.names</code> ) so that they are.
<code>var.names</code>	NULL or a character vector giving the column names for the <code>mvabund</code> object.
<code>neg</code>	character. If FALSE negative values will cause an error message.
<code>na.rm</code>	logical, whether missing values should be removed.
<code>x</code>	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)
```

**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](#). [many1m](#).

## 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 Object*

---

## Description

A residual vs fits plot from a manyany 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. 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.

## 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

[manyany](#)

## 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	manyglm object or manyglm object, typically the result of a call to <code>manyglm</code> or <code>manyglm</code> .
res.type	type of residuals to plot. By default, <code>res.type="pit-norm"</code> uses Dunn-Smyth residuals (Dunn & Smyth 2006), related to the probability integral transform, for <code>manyglm</code> . 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 <code>sub(s.title)</code> otherwise. If <code>NULL</code> , as by default, a possible shortened version of <code>deparse(x\$call)</code> 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 `manyglm`. 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 `manyglm` 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 `manyglm`.

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 `many1m` 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:
solbmabund <- mvabund(solbdat)
plot(solbmabund)

## 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(solbmabund,treatment)

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

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

plot(solbmabund~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(solbmabund~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")

```



**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**

<code>x</code>	a mvabund object, a matrix with multivariate abundance data.
<code>y</code>	a factor or a data.frame with factors, non-factor columns in a data.frame are ignored.
<code>type</code>	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.
<code>main</code>	the title of the plot, see plot.
<code>n.vars</code>	the number of variables to include in the plot.
<code>transformation</code>	an optional transformation, "no" = untransformed, "sqrt"=square root transformed, "log" (default)=log(Y/min+1) transformed, "sqrt4" =4th root transformed.
<code>legend</code>	logical, whether a legend should be added to the plot.
<code>...</code>	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  $\sigma^2$ : 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:

<code>fit</code>	vector or matrix as above
<code>se.fit</code>	a matrix with the standard errors of the predicted means
<code>residual.scale</code>	vector or matrix as a vector of residual standard deviations
<code>df</code>	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)
```

---

residuals.manyglm      *Residuals for MANYGLM or MANYANY Fits*

---

## Description

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

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](#).

**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 $N \leq 20$ , leave-one-out is used $refs=1:N$ , if $N \leq 40$ , 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.

<code>only.ridge</code>	logical, whether only the ridge Parameters should be passed back or additionally the Cross Validation penalised likelihood.
<code>doPlot</code>	logical, whether a plot of $-2\log L$ vs a candidate for the ridge parameter should be drawn.
<code>col</code>	color of Plot symbols.
<code>type</code>	type of Plot symbols.
<code>...</code>	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	<i>Calculate a shift for plotting overlapping points</i>
-------------	--

---

**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.,
Rhabdodemia_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 (*Lycosidae*, *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	<i>Summarizing Multivariate Generalized Linear Model Fits for Abundance Data</i>
-----------------	--

---

## 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 <a href="#">manyglm</a> .
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 <a href="#">symnum</a> ) 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 bootstrap) 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 zero parameters, so is not recommended for multi-sample tests, where such situations are common.

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

<code>call</code>	the component from object.
<code>terms</code>	the terms object used.
<code>family</code>	the component from object.
<code>deviance</code>	the component from object.
<code>aic</code>	Akaike's <i>An Information Criterion</i> , 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).
<code>df.residual</code>	the component from object.
<code>null.deviance</code>	the component from object.
<code>df.null</code>	the component from object.
<code>dev11</code>	minus twice the maximized log-likelihood
<code>iter</code>	the number of iterations that were used in <code>manyglm</code> 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 <a href="#">manylm</a> .
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 be used 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 $R^2 = \text{tr}(\text{SST}^{-1}\text{SSR})/p$ "v" = vector $R^2 = \text{det}(\text{SSR})/\text{det}(\text{SST})$ "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 <a href="#">print.summary.glm</a> 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

<code>call</code>	the component from object.
<code>terms</code>	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 additional 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 additional 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 variable.

**block** A four-level factor variable.

**Details**

The count data (number of each Meiobenthos species in each sample) were collected in a spatially 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, Nannolaimoides.A,  
Nannolaimoides.B, Neochromadora.A, Neochromadora.B,

Odontophora, Oncholaimus, Qnvx,  
 Paracanthochus, 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 community 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, Acropora 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, Hydnoportia rigida, Leptastrea purpurea, Leptastrea pruinosa, Cyphastrea serailia, Millepora platyphylla, Millepora dichotoma, Millepora intricata, 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 1982/3 El Nino in the Thousand Islands, Indonesia, *Coral Reefs* **8**, 171-179.

## Examples

```
require(graphics)

data(tikus)

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

plot(foo)
```

---

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**
  - many`lm.fit`, 34
- \*Topic **classes**
  - mvabund, 37
  - unabund, 74
- \*Topic **datasets**
  - solberg, 58
  - spider, 59
  - Tasmania, 71
  - tikus, 72
- \*Topic **documentation**
  - mvabund-package, 2
- \*Topic **dplot**
  - shiftpoints, 57
- \*Topic **hplot**
  - boxplot.mvabund, 17
  - meanvar.plot, 35
  - plot.manyany, 40
  - plot.many`lm`, 41
  - plot.mvabund, 44
  - plotMvaFactor, 48
- \*Topic **htest**
  - best.r.sq, 16
- \*Topic **manip**
  - formulaUnimva, 22
  - unabund, 74
- \*Topic **methods**
  - predict.many`lm`, 52
- \*Topic **models**
  - anova.manyany, 5
  - anova.many`glm`, 8
  - anova.many`lm`, 12
  - best.r.sq, 16
  - deviance.many`lm`, 20
  - extend.x.formula, 21
  - formulaUnimva, 22
  - logLik.many`lm`, 23
  - manyany, 24
  - many`glm`, 26
  - many`lm`, 31
  - mvformula, 39
  - predict.many`glm`, 50
  - residuals.many`glm`, 54
  - ridgeParamEst, 55
  - summary.many`glm`, 61
  - summary.many`lm`, 66
- \*Topic **multivariate**
  - anova.manyany, 5
  - anova.many`glm`, 8
  - anova.many`lm`, 12
  - best.r.sq, 16
  - deviance.many`lm`, 20
  - formulaUnimva, 22
  - logLik.many`lm`, 23
  - manyany, 24
  - many`glm`, 26
  - many`lm`, 31
  - many`lm.fit`, 34
  - mvabund, 37
  - mvformula, 39
  - plot.manyany, 40
  - plot.many`lm`, 41
  - predict.many`glm`, 50
  - predict.many`lm`, 52
  - residuals.many`glm`, 54
  - summary.many`glm`, 61
  - summary.many`lm`, 66
- \*Topic **regression**
  - anova.manyany, 5
  - anova.many`glm`, 8
  - anova.many`lm`, 12
  - best.r.sq, 16
  - manyany, 24
  - many`glm`, 26
  - many`lm`, 31
  - many`lm.fit`, 34
  - plot.manyany, 40
  - plot.many`lm`, 41

- predict.manyglm, 50
  - residuals.manyglm, 54
  - summary.manyglm, 61
  - summary.manylm, 66
- \*Topic **sysdata**
  - unabund, 74
- anova.manyany, 5, 25, 26
- anova.manyglm, 3, 6, 7, 8, 28, 30, 62, 65
- anova.manylm, 3, 4, 12, 32, 33, 70
- as.data.frame, 27
- as.mvabund, 74
- as.mvabund (mvabund), 37
- as.mvformula (mvformula), 39
- best.r.sq, 16, 22
- boxplot.mvabund, 3, 4, 17, 47
- boxplot.mvformula (boxplot.mvabund), 17
- coefficients, 28
- deviance.manylm, 20
- extend.x.formula, 21
- family, 24–26, 29
- formula, 24–26, 28
- formulaUnimva, 22, 22
- is.mvabund, 74
- is.mvabund (mvabund), 37
- is.mvformula (mvformula), 39
- jitter, 57
- lm, 25, 28, 53
- logLik.manylm, 23
- manyany, 5–7, 24, 40, 41
- manyglm, 3, 8–11, 25, 26, 51, 54, 61–63, 65
- manylm, 3, 4, 12, 15, 21, 31, 34, 35, 39, 42, 43, 53, 56, 66, 70
- manylm.fit, 32, 34
- manylm.wfit, 32
- manylm.wfit (manylm.fit), 34
- meanvar.plot, 3, 4, 35, 47
- meanvar.plot, data.frame-method (meanvar.plot), 35
- meanvar.plot, formula-method (meanvar.plot), 35
- meanvar.plot, matrix-method (meanvar.plot), 35
- meanvar.plot, mvabund-method (meanvar.plot), 35
- meanvar.plot, mvformula-method (meanvar.plot), 35
- meanvar.plot.mvabund (meanvar.plot), 35
- meanvar.plot.mvformula (meanvar.plot), 35
- mvabund, 3, 4, 23, 37, 39, 74
- mvabund-class (mvabund), 37
- mvabund-package, 2
- mvformula, 3, 4, 22, 23, 38, 39
- mvformula-class (mvformula), 39
- na.exclude, 27
- na.fail, 27
- na.omit, 27
- napredict, 51, 52
- options, 27
- plot.manyany, 6, 26, 40
- plot.manyglm, 3, 9, 47, 62
- plot.manyglm (plot.manylm), 41
- plot.manylm, 3, 4, 15, 33, 41, 47, 70
- plot.mvabund, 2, 4, 20, 36, 38, 44, 50, 57
- plot.mvformula, 22, 36, 57
- plot.mvformula (plot.mvabund), 44
- plotMvaFactor, 3, 4, 48
- predict.manyglm, 50
- predict.manylm, 52
- print.anova.manyany (anova.manyany), 5
- print.anova.manyglm (anova.manyglm), 8
- print.anova.manylm (anova.manylm), 12
- print.summary.glm, 8, 62, 67
- print.summary.manyglm (summary.manyglm), 61
- print.summary.manylm (summary.manylm), 66
- residuals, 29
- residuals.manyany, 26
- residuals.manyany (residuals.manyglm), 54
- residuals.manyglm, 40, 54
- ridgeParamEst, 10, 14, 28, 55, 63, 68
- sample, 5

shiftpoints, [57](#)  
solberg, [3](#), [58](#)  
spider, [3](#), [59](#)  
summary.manyglm, [3](#), [9](#), [11](#), [28](#), [30](#), [61](#)  
summary.manylm, [3](#), [4](#), [15](#), [32](#), [33](#), [66](#)  
symnum, [61](#)

Tasmania, [3](#), [71](#)  
terms, [29](#)  
tikus, [3](#), [72](#)

unabund, [38](#), [74](#), [74](#)