Package 'mixlm'

August 8, 2023

Type Package

Title Mixed Model ANOVA and Statistics for Education

Version 1.3.0

Date 2023-08-08

Maintainer Kristian Hovde Liland <kristian.liland@nmbu.no>

Encoding UTF-8

Description The main functions perform mixed models analysis by least squares or REML by adding the function r() to formulas of lm() and glm(). A collection of text-book statistics for higher education is also included, e.g. modifications of the functions lm(), glm() and associated summaries from the package 'stats'.

Imports car, pls, multcomp, pracma, leaps

Suggests lme4

License GPL (>= 2)

URL https://github.com/khliland/mixlm/

BugReports https://github.com/khliland/mixlm/issues/

NeedsCompilation no

Author Kristian Hovde Liland [aut, cre],
Solve Sæbø, [ctb],
R-Core [ctb]

Repository CRAN

Date/Publication 2023-08-08 14:00:07 UTC

R topics documented:

`Anova.lmm` *Analysis of variance with SS type II or III (including mixed models).*

Description

Replacement for `Anova.lm` in package `car`. This version adds support for random effects when needed.

Usage

```
## S3 method for class 'lmm'
Anova(mod, ...)
```

Arguments

`mod`	`lm`, `aov`, `glm`, `multinom`, `polr` `mlm`, `coxph`, `lme`, `mer`, `svyglm` or other suitable model object.
`...`	do not use.

Value

Returns appropriate analysis of variance or halts if unsupported input is detected.

Author(s)

John Fox `<jfox@mcmaster.ca>`. Extended by Kristian Hovde Liland.

See Also

Anova, print.AnovaMix, AnovaMix, lm

Examples

```
dataset <- data.frame(y = rnorm(8),
  x = factor(c(rep(1,4),rep(0,4))),
  z = factor(rep(c(1,0),4)))
mixlm <- lm(y~x*r(z),
  data = dataset)
Anova(mixlm, type="III")
```

---

anova.lmm *Analysis of variance (sequential SS)*

Description

Wrapper for anova.lm in package stats that halts execution if unsupported input is detected.

Usage

```
## S3 method for class 'lmm'
anova(object, ...)
```

Arguments

object	object fitted by lm, lmer or similar.
...	possible additional argument to underlying functions.

Value

Returns appropriate analysis of variance or halts if unsupported input is detected.

Author(s)

Kristian Hovde Liland

See Also

lm

Examples

```
mixlm <- lm(y~x*r(z),
data = data.frame(y = rnorm(8),
  x = factor(c(rep(1,4),rep(0,4))),
  z = factor(rep(c(1,0),4))))
anova(mixlm)
```

AnovaMix *Mixed model least squares analysis of variance (mixed ANOVA).*

Description

Uses output from lm() in mixlm package to compute ANOVA table, variance components and errors.

Usage

```
AnovaMix(object, SStype)
```

Arguments

object	object fitted by lm (mixlm package) containing at least one random effect.
SStype	type of sums-of-squares (I/II/III) for Analysis of Variance.

Details

AnovaMix can either be invoked directly or through the Anova() function (with type III error).

Value

lm	linear model fitted by lm in package mixlm.
anova	ANOVA table.
err.terms	list of denominator information for F tests.
denom.df	numeric of denominator degrees of freedom for F tests.
restricted	logical indicating if ANOVA used restricted modelling.
exp.mean.sq	character containing expected mean squares.
var.comps	numeric containing variance components.
random.effects	character containing the random effects.
ind.randoms	numeric with indices of random effects in the model.
formula.text	character containing all effects of the model.

Note

Only balanced models are fully supported.

Author(s)

Kristian Hovde Liland

See Also

print.AnovaMix, Anova, lm

Examples

```
mydata <- data.frame(y = rnorm(12),
  x = factor(c(rep(2,4),rep(1,4),rep(0,4))),
  z = factor(rep(c(1,0),6)))
mixlm <- lm(y~x*r(z),
data = mydata)
Anova(mixlm,type="III")
```

---

anova_reg *Analysis of variance for regression.*

Description

Summarizes all effects in one.

Usage

```
anova_reg(lm.object)
```

Arguments

`lm.object`	an object of class `lm`.

Value

Returns data.frame containing analysis of variance

Author(s)

Kristian Hovde Liland

Examples

```
anova_reg(lm(y~x, data=data.frame(y=1:4,x=rnorm(4))))
```

`best.subsets` *F-test based best subset selection.*

Description

Adaptation of existing methods based on AIC/BIC.

Usage

```
best.subsets(model, nbest = 5, nvmax, digits, force.in = "NULL")
```

Arguments

`model`	object class `lm` to select effects from.
`nbest`	`numeric` indicating number of models to report of each size.
`nvmax`	`numeric` maximum size of subsets to examine.
`digits`	`numeric` giving number of digits in format of output.
`force.in`	`character` vector indicating effects to keep in all models.

Details

F-based versions of built in subset method.

Value

No return, only print.

Author(s)

Kristian Hovde Liland

Examples

```
data <- data.frame(y = rnorm(8),
   x = factor(c('a','a','a','a','b','b','b','b')),
   z = factor(c('a','a','b','b','a','a','b','b')))
mod <- lm(y ~ x + z, data=data)
best.subsets(mod)
```

CIgrandMean *Confidence interval for the grand mean of a linear model*

Description

This function estimates the confidence interval for the grand mean of a balanced linear (mixed) model.

Usage

```
CIgrandMean(object, alpha = 0.05)
## S3 method for class 'CIgm'
print(x, ...)
```

Arguments

object	An lm object possibly containing random effects.
alpha	A scalar significance level for the confidence interval.
x	An object returned from CIgrandMean.
...	Additional arguments (not used).

Details

This implementation is only valid for models containing no continuous effects and only balanced data.

Value

CIgrandMean returns a vector of interval endpoints and center. print.CIgm has no return.

Author(s)

Kristian Hovde Liland

References

Suggestions are welcome.

Examples

```
set.seed(42)
dataset  <- data.frame(y=rnorm(8), x=factor(c(rep(1,4),rep(0,4))), z=factor(rep(c(1,0),4)))
mixlm <- lm(y~x*r(z), data = dataset)
CIgrandMean(mixlm)
```

confusion *Confusion matrix.*

Description

Computes the confusion matrix of a classification result.

Usage

```
confusion(true, predicted)
```

Arguments

true	true classes.
predicted	predicted classes.

Details

This is a pure print function.

Examples

```
true <- c('a','a','b','b','c','c')
predicted <- c('a','c','b','b','a','c')
confusion(true, predicted)
```

contr.weighted *Contrast matrix for weighted effect coding*

Description

Weighted contrast coding for linear models.

Usage

```
contr.weighted(x, base)
```

Arguments

x	factor for which a contrast matrix should be made.
base	factor level used as basis for contrast coding. Default is the (first) level with maximum frequency.

Details

Different from the contrasts made throught the `stats` package functions this contrast requires a full `factor` vector as input rather than its respective `levels` as weights are computed from the frequencies of the `factor` `levels`.

Value

A `matrix` with `n` rows and `n-1` values.

Note

`contr.weighted` cannot be used directly as a replacement for other contrasts by name, but must be used via contrasts matrix computations.

Author(s)

Kristian Hovde Liland

References

Nieuwenhuis, R.; Grotenhuis, M.; Pelzer, B. Weighted Effect Coding for Observational Data with wec. R. J. 2017, 9, 477–485.

See Also

`lm`

Examples

```
balanced   <- factor(c(rep("A", 3), rep("B", 3), rep("C", 3)))
unbalanced <- factor(c(rep("A", 3), rep("B", 3), rep("C", 2)))
# Weighted coding when applied to balanced data
contr.weighted(balanced)
# Weighted coding when applied to unbalanced data (default base level)
contr.weighted(unbalanced)
# Weighted coding when applied to unbalanced data (base level = "C")
contr.weighted(unbalanced, "C")
```

`effect.labels` *Create new effect labels for* `lm`

Description

Alternative notation of effect labels including levels.

Usage

```
effect.labels(t, data)
```

Arguments

t	Terms object.
data	Corresponding model.matrix.

Value

names	Character vector of effect labels.

Author(s)

Kristian Hovde Liland

---

forward *F-test based model effect selection for linear models.*

Description

Adaptation of existing methods based on AIC/BIC.

Usage

```
forward(model, alpha = 0.2, full = FALSE, force.in)
backward(model, alpha = 0.2, full = FALSE, hierarchy = TRUE, force.in)
stepWise(model, alpha.enter = 0.15, alpha.remove = 0.15, full = FALSE)
stepWiseBack(model, alpha.remove = 0.15, alpha.enter = 0.15, full = FALSE)
wideForward(formula, data, alpha = 0.2, force.in = NULL)
```

Arguments

model	object class lm to select effects from.
formula	formula specifying all possible effects.
data	data.frame corresponding to formula.
alpha	numeric p-value cut-off for inclusion/exclusion.
full	logical indicating extended output of forward/backward selection.
force.in	character vector indicating effects to keep in all models.
alpha.enter	numeric p-value cut-off for inclusion.
alpha.remove	numeric p-value cut-off for exclusion.
hierarchy	logical indicating if hierarchy should be forced in backward selection.

Details

F-based versions of built in stepwise methods.

Value

The final linear model after selection is returned.

Author(s)

Kristian Hovde Liland

Examples

```
set.seed(0)
data <- data.frame(y = rnorm(8),
   x = factor(c('a','a','a','a','b','b','b','b')),
   z = factor(c('a','a','b','b','a','a','b','b')))
mod <- lm(y ~ x + z, data=data)
forward(mod)
backward(mod)
stepWise(mod)
stepWiseBack(mod)

# Forward selection for wide matrices (large number of predictors)
set.seed(0)
mydata <- data.frame(y = rnorm(6), X = matrix(rnorm(60),6,10))
fs <- wideForward(y ~ ., mydata)
print(fs)
```

`fparse` *Effects of formulas.*

Description

Extracts all effects from a formula, even though inside functions or interactions.

Usage

```
fparse(f)
```

Arguments

`f`	`formula` to be parsed.

Value

Returns a `character` vector containing all effects.

Author(s)

Bjørn-Helge Mevik

See Also

rparse

Examples

```
f <- formula(y~x*r(z))
fparse(f)
```

---

glm *Fitting Generalized Linear Models*

Description

glm is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution. The version of glm supplied by the package mixlm parses to glmer for mixed modelling.

Usage

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL, REML = TRUE, ...)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
family	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
start	starting values for the parameters in the linear predictor.

etastart	starting values for the linear predictor.
mustart	starting values for the vector of means.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
control	a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control.
model	a logical value indicating whether *model frame* should be included as a component of the returned value.
method	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS): the alternative "model.frame" returns the model frame and does no fitting. User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If specified as a character string it is looked up from within the **stats** namespace.
x, y	For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
REML	is used to invoke restricted maximum likelihood (TRUE) or maximum likelihood (FALSE) estimation instead of least squares.
...	For glm: arguments to be used to form the default control argument if it is not supplied directly. For weights: further arguments passed to or from other methods.

Details

A typical predictor has the form response ~ terms where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response. For binomial and quasibinomial families the response can also be specified as a factor (when the first level denotes failure and all others success) or as a two-column matrix with the columns giving the numbers of successes and failures. A terms specification of the form first + second indicates all the terms in first together with all the terms in second with any duplicates removed.

A specification of the form first:second indicates the the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the *cross* of first and second. This is the same as first + second + first:second.

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.

Non-NULL weights can be used to indicate that different observations have different dispersions (with the values in weights being inversely proportional to the dispersions); or equivalently, when

the elements of weights are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations. For a binomial GLM prior weights are used to give the number of trials when the response is the proportion of successes: they would rarely be used for a Poisson GLM.

glm.fit is the workhorse function: it is not normally called directly but can be more efficient where the response vector and design matrix have already been calculated.

If more than one of etastart, start and mustart is specified, the first in the list will be used. It is often advisable to supply starting values for a quasi family, and also for families with unusual links such as gaussian("log").

All of weights, subset, offset, etastart and mustart are evaluated in the same way as variables in formula, that is first in data and then in the environment of formula.

For the background to warning messages about 'fitted probabilities numerically 0 or 1 occurred' for binomial GLMs, see Venables & Ripley (2002, pp. 197–8).

Value

glm returns an object of class inheriting from "glm" which inherits from the class "lm". See later in this section. If a non-standard method is used, the object will also inherit from the class (if any) returned by that function.

The function summary (i.e., summary.glm) can be used to obtain or print a summary of the results and the function anova (i.e., anova.glm) to produce an analysis of variance table.

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

weights extracts a vector of weights, one for each case in the fit (after subsetting and na.action).

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

coefficients	a named vector of coefficients
residuals	the *working* residuals, that is the residuals in the final iteration of the IWLS fit. Since cases with zero weights are omitted, their working residuals are NA.
fitted.values	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
rank	the numeric rank of the fitted linear model.
family	the family object used.
linear.predictors	the linear fit on link scale.
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	A version of Akaike's *An Information Criterion*, minus twice the maximized log-likelihood plus twice the number of parameters, computed by the aic component of the family. For binomial and Poison families the dispersion is fixed at one and the number of parameters is the number of coefficients. For gaussian, Gamma and inverse gaussian families the dispersion is estimated from the residual deviance, and the number of parameters is the number of coefficients plus one. For a gaussian family the MLE of the dispersion is used so this is a valid value of AIC, but for Gamma and inverse gaussian families it is not. For families fitted by quasi-likelihood the value is NA.

null.deviance	The deviance for the null model, comparable with deviance. The null model will include the offset, and an intercept if there is one in the model. Note that this will be incorrect if the link function depends on the data other than through the fitted mean: specify a zero offset to force a correct calculation.
iter	the number of iterations of IWLS used.
weights	the *working* weights, that is the weights in the final iteration of the IWLS fit.
prior.weights	the weights initially supplied, a vector of 1s if none were.
df.residual	the residual degrees of freedom.
df.null	the residual degrees of freedom for the null model.
y	if requested (the default) the y vector used. (It is a vector even for a binomial model.)
x	if requested, the model matrix.
model	if requested (the default), the model frame.
converged	logical. Was the IWLS algorithm judged to have converged?
boundary	logical. Is the fitted value on the boundary of the attainable values?
call	the matched call.
formula	the formula supplied.
terms	the terms object used.
data	the data argument.
offset	the offset vector used.
control	the value of the control argument used.
method	the name of the fitter function used, currently always "glm.fit".
contrasts	(where relevant) the contrasts used.
xlevels	(where relevant) a record of the levels of the factors used in fitting.
na.action	(where relevant) information returned by model.frame on the special handling of NAs.

In addition, non-empty fits will have components qr, R and effects relating to the final weighted linear fit.

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.

If a binomial glm model was specified by giving a two-column response, the weights returned by prior.weights are the total numbers of cases (factored by the supplied case weights) and the component y of the result is the proportion of successes.

Fitting functions

The argument method serves two purposes. One is to allow the model frame to be recreated with no fitting. The other is to allow the default fitting function glm.fit to be replaced by a function which takes the same arguments and uses a different fitting algorithm. If glm.fit is supplied as a character string it is used to search for a function of that name, starting in the **stats** namespace.

The class of the object return by the fitter (if any) will be prepended to the class returned by glm.

Author(s)

The original R implementation of `glm` was written by Simon Davies working for Ross Ihaka at the University of Auckland, but has since been extensively re-written by members of the R Core team.

The design was inspired by the S function of the same name described in Hastie & Pregibon (1992).

Mixed model additions by Kristian Hovde Liland.

References

Dobson, A. J. (1990) *An Introduction to Generalized Linear Models.* London: Chapman and Hall.

Hastie, T. J. and Pregibon, D. (1992) *Generalized linear models.* Chapter 6 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models.* London: Chapman and Hall.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S.* New York: Springer.

See Also

`anova.glm`, `summary.glm`, etc. for `glm` methods, and the generic functions `anova`, `summary`, `effects`, `fitted.values`, and `residuals`.

`lm` for non-generalized *linear* models (which SAS calls GLMs, for 'general' linear models).

`loglin` and `loglm` (package MASS) for fitting log-linear models (which binomial and Poisson GLMs are) to contingency tables.

`bigglm` in package biglm for an alternative way to fit GLMs to large datasets (especially those with many cases).

`esoph`, `infert` and `predict.glm` have examples of fitting binomial glms.

Examples

```
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))
glm.D93 <- glm(counts ~ outcome + treatment, family=poisson())
anova(glm.D93)

# A Gamma example, from McCullagh & Nelder (1989, pp. 300-2)
clotting <- data.frame(
    u = c(5,10,15,20,30,40,60,80,100),
    lot1 = c(118,58,42,35,27,25,21,19,18),
    lot2 = c(69,35,26,21,18,16,13,12,12))
summary(glm(lot1 ~ log(u), data=clotting, family=Gamma))
summary(glm(lot2 ~ log(u), data=clotting, family=Gamma))

# Mixed model example
dataset  <- data.frame(y=rnorm(8), x=factor(c(rep(1,4),rep(0,4))), z=factor(rep(c(1,0),4)))
if(require(lme4)){
  GLM       <- glm(y  ~ x+r(z), family=gaussian(identity), data=dataset)
```

```
  summary(GLM)
  logLik(GLM)
  Anova(GLM,type=3)
}

## Not run:
## for an example of the use of a terms object as a formula
demo(glm.vr)

## End(Not run)
```

`is.balanced` *Balance cheking of models.*

Description

Checks if models have balanced data.

Usage

```
is.balanced(object)
```

Arguments

`object`	fitted model that includes `variables` attribute and `model` slot.

Value

Returns TRUE if balanced, FALSE if not.

Author(s)

Kristian Hovde Liland

Examples

```
mixlm <- lm(y~x*r(z),
    data = data.frame(y = rnorm(8),
  x = factor(c(rep(1,4),rep(0,4))),
  z = factor(rep(c(1,0),4))))
is.balanced(mixlm)
```

lm *Fitting Linear Models*

Description

lm is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although aov may provide a more convenient interface for these). The version distributed through the package mixlm extends the capabilities with balanced mixture models and lmer interfacing. Random effects are indicated by wrapping their formula entries in r(). Also, effect level names are kept in printing.

Usage

```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = "contr.sum", offset,
   unrestricted = TRUE, REML = NULL, ...)
```

Arguments

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

contrasts	character indicating which coding should be applied to all factors or an optional list. See the contrasts.arg of model.matrix.default. Defaults to "contr.sum". See Details for more information.
offset	this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one are specified their sum is used. See model.offset.
unrestricted	additional argument for switching between unrestricted and restricted models if including random variables.
REML	is used to invoke restricted maximum likelihood (TRUE) or maximum likelihood (FALSE) estimation instead of least squares.
...	additional arguments to be passed to the low level regression fitting functions (see below).

Details

Models for lm are specified symbolically. A typical model has the form response ~ terms where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response. A terms specification of the form first + second indicates all the terms in first together with all the terms in second with duplicates removed. A specification of the form first:second indicates the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the *cross* of first and second. This is the same as first + second + first:second.

If the formula includes an offset, this is evaluated and subtracted from the response.

If response is a matrix a linear model is fitted separately by least-squares to each column of the matrix.

See model.matrix for some further details. The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula (see aov and demo(glm.vr) for an example).

A formula has an implied intercept term. To remove this use either y ~ x - 1 or y ~ 0 + x. See formula for more details of allowed formulae.

Non-NULL weights can be used to indicate that different observations have different variances (with the values in weights being inversely proportional to the variances); or equivalently, when the elements of weights are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations (including the case that there are w_i observations equal to y_i and the data have been summarized).

lm calls the lower level functions lm.fit, etc, see below, 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.

The contrasts argument is applied when the lm model is fitted. In additional to the standard contrasts from the stats package, one can choose weighted coding: contr.weighted which balances unbalanced data through factor weighting. Different from the original version of lm, a single contrast can be indicated to automatically be applied to all factors.

Value

`lm` returns an object of `class` `c("lmm","lm")` or for multiple responses of class `c("mlm", "lm")`.

The functions `summary` and `anova` are used to obtain and print a summary and analysis of variance table of the results. The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` extract various useful features of the value returned by `lm`.

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

`coefficients`	a named vector of coefficients
`residuals`	the residuals, that is response minus fitted values.
`fitted.values`	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.
`call`	the matched call.
`terms`	the `terms` object used.
`contrasts`	(only where relevant) the contrasts used.
`xlevels`	(only where relevant) a record of the levels of the factors used in fitting.
`offset`	the offset used (missing if none were used).
`y`	if requested, the response used.
`x`	if requested, the model matrix used.
`model`	if requested (the default), the model frame used.
`na.action`	(where relevant) information returned by `model.frame` on the special handling of NAs.

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

And models containing random effect will contain `random` having additional information about the model.

Using time series

Considerable care is needed when using `lm` with time series.

Unless `na.action = NULL`, the time series attributes are stripped from the variables before the regression is done. (This is necessary as omitting NAs would invalidate the time series attributes, and if NAs are omitted in the middle of the series the result would no longer be a regular time series.)

Even if the time series attributes are retained, they are not used to line up series, so that the time shift of a lagged or differenced regressor would be ignored. It is good practice to prepare a `data` argument by `ts.intersect(..., dframe = TRUE)`, then apply a suitable `na.action` to that data frame and call `lm` with `na.action = NULL` so that residuals and fitted values are time series.

Note

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

Author(s)

The design was inspired by the S function of the same name described in Chambers (1992). The implementation of model formula by Ross Ihaka was based on Wilkinson & Rogers (1973). Mixed model extensions by Kristian Hovde Liland.

References

Chambers, J. M. (1992) *Linear models.* Chapter 4 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

Wilkinson, G. N. and Rogers, C. E. (1973) Symbolic descriptions of factorial models for analysis of variance. *Applied Statistics*, **22**, 392–9.

See Also

`summary.lmm` for summaries and `anova.lmm` for the ANOVA table; `aov` for a different interface.

The generic functions `coef`, `effects`, `residuals`, `fitted`, `vcov`.

`predict.lm` (via `predict`) for prediction, including confidence and prediction intervals; `confint` for confidence intervals of *parameters*.

`lm.influence` for regression diagnostics, and `glm` for **generalized** linear models.

The underlying low level functions, `lm.fit` for plain, and `lm.wfit` for weighted regression fitting.

More `lm()` examples are available e.g., in `anscombe`, `attitude`, `freeny`, `LifeCycleSavings`, `longley`, `stackloss`, `swiss`.

`biglm` in package **biglm** for an alternative way to fit linear models to large datasets (especially those with many cases).

`print.AnovaMix`, `AnovaMix`, `Anova`

Examples

```
require(graphics)

## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight ~ group)
lm.D90 <- lm(weight ~ group - 1) # omitting intercept
anova(lm.D9)
summary(lm.D90)

opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))
plot(lm.D9, las = 1)      # Residuals, Fitted, ...
par(opar)

# Linear mixed model
dataset  <- data.frame(y=rnorm(8), x=factor(c(rep(1,4),rep(0,4))), z=factor(rep(c(1,0),4)))
mixlm <- lm(y~x*r(z), data = dataset)
```

```
Anova(mixlm,type="III")

### less simple examples in "See Also" above
```

---

plotprops *Property plots for relevant component analysis*

Description

Plot summary of relevant component analysis.

Usage

```
plotprops(Y, X, doscaleX = FALSE, docenterX = TRUE, ncomp, subset)
```

Arguments

Y	Response matrix.
X	Predictor matrix.
doscaleX	Standardize predictors.
docenterX	Center predictors.
ncomp	Number of singular values and eigenvalues to extract.
subset	Subset of predictor and response.

Value

Only plotting.

Author(s)

Solve Sæbø

References

Helland, I.S. & T. Almøy (1994) Comparison of prediction methods when only a few components are relevant. JASA 89, 583-591.

Examples

```
X <- matrix(rnorm(100),20,5)
Y <- matrix(rnorm(20),20,1)
plotprops(Y, X, doscaleX = FALSE, docenterX = TRUE, 5)
```

PRESS *Prediction fits*

Description

Various summaries of predictions and PRESS residuals.

Usage

```
R2pred(object = NULL)
RMSEP(object)
rmsep(object)
PRESS(object = NULL)
PRESS.res(object = NULL, ncomp = NULL)
PRESS.pred(object = NULL, ncomp = NULL)
```

Arguments

object	a fitted model of type lm or mvr.
ncomp	number of components to use with mvr (optional).

Details

Predictions are extracted and summaries/residuals are computed.

Value

Returns either an object of summaries or residuals.

Author(s)

Kristian Hovde Liland

Examples

```
data <- data.frame(y = rnorm(8),
   x = factor(c('a','a','a','a','b','b','b','b')),
   z = factor(c('a','a','b','b','a','a','b','b')))
mod <- lm(y ~ x + z, data=data)
RMSEP(mod)
rmsep(mod) # Alias to distinguish it from pls::RMSEP
R2pred(mod)
PRESS(mod)
PRESS.res(mod)
PRESS.pred(mod)
```

`print.AnovaMix` *Print method for objects of class(AnovaMix)*

Description

Prints relevant information like the ANOVA table, variance components and errors.

Usage

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

Arguments

`x`	`AnovaMix` object to be printed.
`...`	Additional arguments (not supported yet).

Note

Only balanced models are fully supported.

Author(s)

Kristian Hovde Liland

See Also

`AnovaMix`, `lm`, `Anova`

Examples

```
mixlm <- lm(y~x*r(z),
data = data.frame(y = rnorm(8),
  x = factor(c(rep(1,4),rep(0,4))),
  z = factor(rep(c(1,0),4))))
Anova(mixlm,type="III")
```

`print.summary.lmm` *Summarizing Linear Model Fits*

Description

`summary` method for class `"lmm"`.

Usage

```
## S3 method for class 'summary.lmm'
print(x, digits = max(3, getOption("digits") - 3),
      symbolic.cor = x$symbolic.cor,
      signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

`x`	an object of class `"summary.lmm"`, usually, a result of a call to `summary.lmm`.
`digits`	the number of significant digits to use when printing.
`symbolic.cor`	logical. If `TRUE`, print the correlations in a symbolic form (see `symnum`) rather than as numbers.
`signif.stars`	logical. If `TRUE`, 'significance stars' are printed for each coefficient.
`...`	further arguments passed to or from other methods.

Details

This adaptation of `print.summary.lm` from package `stats` slightly alters the output to better conform with text-book notation.

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

Correlations are printed to two decimal places (or symbolically): to see the actual correlations print `summary(object)$correlation` directly.

Value

The function `summary.lm` computes and returns a list of summary statistics of the fitted linear model given in `object`, using the components (list elements) `"call"` and `"terms"` from its argument, plus

`residuals`	the *weighted* residuals, the usual residuals rescaled by the square root of the weights specified in the call to `lm`.
`coefficients`	a $p \times 4$ matrix with columns for the estimated coefficient, its standard error, t-statistic and corresponding (two-sided) p-value. Aliased coefficients are omitted.
`aliased`	named logical vector showing if the original coefficients are aliased.

`sigma` the square root of the estimated variance of the random error

$$\hat{\sigma}^2 = \frac{1}{n-p}\sum_i w_i R_i^2,$$

where R_i is the i-th residual, `residuals[i]`.

`df` degrees of freedom, a 3-vector $(p, n-p, p*)$, the last being the number of non-aliased coefficients.

`fstatistic` (for models including non-intercept terms) a 3-vector with the value of the F-statistic with its numerator and denominator degrees of freedom.

`r.squared` R^2, the 'fraction of variance explained by the model',

$$R^2 = 1 - \frac{\sum_i R_i^2}{\sum_i (y_i - y^*)^2},$$

where y^* is the mean of y_i if there is an intercept and zero otherwise.

`adj.r.squared` the above R^2 statistic '*adjusted*', penalizing for higher p.

`cov.unscaled` a $p \times p$ matrix of (unscaled) covariances of the $\hat{\beta}_j$, $j = 1, \ldots, p$.

`correlation` the correlation matrix corresponding to the above `cov.unscaled`, if `correlation = TRUE` is specified.

`symbolic.cor` (only if `correlation` is true.) The value of the argument `symbolic.cor`.

`na.action` from `object`, if present there.

See Also

The model fitting function `lm`, `summary`.

Function `coef` will extract the matrix of coefficients with standard errors, t-statistics and p-values.

Examples

```
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
sld90 <- summary(lm.D90 <- lm(weight ~ group -1))# omitting intercept
sld90
```

`prop.test.ordinary` *Test of Equal or Given Proportions in text-book version*

Description

This adaptation of `prop.test` from package `stats` strips the test down to a text-book version.

`prop.test` can be used for testing the null that the proportions (probabilities of success) in several groups are the same, or that they equal certain given values.

Usage

```
prop.test.ordinary(x, n, p = NULL,
          alternative = c("two.sided", "less", "greater"),
          conf.level = 0.95, correct = TRUE, pooled = TRUE)
```

Arguments

x	a vector of counts of successes, a one-dimensional table with two entries, or a two-dimensional table (or matrix) with 2 columns, giving the counts of successes and failures, respectively.
n	a vector of counts of trials; ignored if x is a matrix or a table.
p	a vector of probabilities of success. The length of p must be the same as the number of groups specified by x, and its elements must be greater than 0 and less than 1.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter. Only used for testing the null that a single proportion equals a given value, or that two proportions are equal; ignored otherwise.
conf.level	confidence level of the returned confidence interval. Must be a single number between 0 and 1. Only used when testing the null that a single proportion equals a given value, or that two proportions are equal; ignored otherwise.
correct	a logical indicating whether Yates' continuity correction should be applied where possible.
pooled	a logical indicating wheter pooled standard deviation should be used..

Details

Only groups with finite numbers of successes and failures are used. Counts of successes and failures must be nonnegative and hence not greater than the corresponding numbers of trials which must be positive. All finite counts should be integers.

If p is NULL and there is more than one group, the null tested is that the proportions in each group are the same. If there are two groups, the alternatives are that the probability of success in the first group is less than, not equal to, or greater than the probability of success in the second group, as specified by alternative. A confidence interval for the difference of proportions with confidence level as specified by conf.level and clipped to $[-1, 1]$ is returned. Continuity correction is used only if it does not exceed the difference of the sample proportions in absolute value. Otherwise, if there are more than 2 groups, the alternative is always "two.sided", the returned confidence interval is NULL, and continuity correction is never used.

If there is only one group, then the null tested is that the underlying probability of success is p, or .5 if p is not given. The alternative is that the probability of success is less than, not equal to, or greater than p or 0.5, respectively, as specified by alternative. A confidence interval for the underlying proportion with confidence level as specified by conf.level and clipped to $[0, 1]$ is returned. Continuity correction is used only if it does not exceed the difference between sample and null proportions in absolute value. The confidence interval is computed by inverting the score test.

Finally, if p is given and there are more than 2 groups, the null tested is that the underlying probabilities of success are those given by p. The alternative is always "two.sided", the returned confidence interval is NULL, and continuity correction is never used.

Value

A list with class "htest" containing the following components:

statistic	the value of Pearson's chi-squared test statistic.
parameter	the degrees of freedom of the approximate chi-squared distribution of the test statistic.
p.value	the p-value of the test.
estimate	a vector with the sample proportions x/n.
conf.int	a confidence interval for the true proportion if there is one group, or for the difference in proportions if there are 2 groups and p is not given, or NULL otherwise. In the cases where it is not NULL, the returned confidence interval has an asymptotic confidence level as specified by conf.level, and is appropriate to the specified alternative hypothesis.
null.value	the value of p if specified by the null, or NULL otherwise.
alternative	a character string describing the alternative.
method	a character string indicating the method used, and whether Yates' continuity correction was applied.
data.name	a character string giving the names of the data.

References

Wilson, E.B. (1927) Probable inference, the law of succession, and statistical inference. *J. Am. Stat. Assoc.*, **22**, 209–212.

Newcombe R.G. (1998) Two-Sided Confidence Intervals for the Single Proportion: Comparison of Seven Methods. *Statistics in Medicine* **17**, 857–872.

Newcombe R.G. (1998) Interval Estimation for the Difference Between Independent Proportions: Comparison of Eleven Methods. *Statistics in Medicine* **17**, 873–890.

See Also

binom.test for an *exact* test of a binomial hypothesis.

Examples

```
heads <- rbinom(1, size=100, prob = .5)
prop.test(heads, 100)          # continuity correction TRUE by default
prop.test(heads, 100, correct = FALSE)

## Data from Fleiss (1981), p. 139.
## H0: The null hypothesis is that the four populations from which
##     the patients were drawn have the same true proportion of smokers.
## A:  The alternative is that this proportion is different in at
##     least one of the populations.

smokers  <- c( 83, 90, 129, 70 )
patients <- c( 86, 93, 136, 82 )
prop.test.ordinary(smokers, patients)
```

rparse *Removes function r() from formulas.*

Description

Removes function r() from formulas. Can also convert to lmer formula.

Usage

```
rparse(f, REML = FALSE)
```

Arguments

f	formula to be stripped of r().
REML	logical indicating if lmer conversion should be done.

Value

Formula without r(), possibly converted to lmer mixed model format.

Author(s)

Kristian Hovde Liland

See Also

fparse

Examples

```
f <- formula(y~x*r(z))
rparse(f)
```

simple.glht *Pairwise comparison with multiple testing compensation.*

Description

Extension of glht from the multcomp package to handle Fisher family-wise error and Bonferroni testing. Create a set of confidence intervals on the differences between the means of the levels of a factor with the specified family-wise probability of coverage. The intervals are based on the Studentized range statistic, Tukey's 'Honest Significant Difference' method, Fisher's family-wise error, or Bonferroni testing.

Usage

```
simple.glht(mod, effect, corr = c("Tukey","Bonferroni","Fisher"),
level = 0.95, df = NULL, ...)
```

Arguments

mod	A fitted model object, usually an lm or glm fit.
effect	A character vector giving the term of the fitted model for which the intervals should be calculated. This can also be an interaction.
corr	A character vector giving the multiple testing correction method. Defaults to Tukey.
level	A numeric value between zero and one giving the family-wise confidence level to use.
df	User supplied number of degrees of freedom. If not supplied or NULL, the default is to extract these from the model.
...	Optional additional arguments. None are used at present.

Details

When comparing the means for the levels of a factor in an analysis of variance, a simple comparison using t-tests will inflate the probability of declaring a significant difference when it is not in fact present. This because the intervals are calculated with a given coverage probability for each interval but the interpretation of the coverage is usually with respect to the entire family of intervals.

John Tukey introduced intervals based on the range of the sample means rather than the individual differences. The intervals returned by this function are based on this Studentized range statistics.

The intervals constructed in this way would only apply exactly to balanced designs where there are the same number of observations made at each level of the factor. This function incorporates an adjustment for sample size that produces sensible intervals for mildly unbalanced designs.

If which specifies non-factor terms these will be dropped with a warning: if no terms are left this is a an error.

Value

An object of classes "simple.glht", "summary.glht" and "glht" containing information to produce confidence intervals, tests and plotting.

There are print, plot and cld methods for class "simple.glht". The plot method does not accept xlab, ylab or main arguments and creates its own values for each plot.

Author(s)

Douglas Bates, extended to mixed effect models by Kristian Hovde Liland.

References

Miller, R. G. (1981) *Simultaneous Statistical Inference*. Springer.

Yandell, B. S. (1997) *Practical Data Analysis for Designed Experiments*. Chapman & Hall.

See Also

aov, qtukey, model.tables, glht in package **multcomp**.

Examples

```
require(graphics)

summary(fm1 <- lm(breaks ~ wool + tension, data = warpbreaks))
simple.glht(fm1, "tension")
plot(simple.glht(fm1, "tension"))
cld(simple.glht(fm1, "tension"))
```

---

spearson *Standardized Pearson residuals*

Description

Standardized Pearson residuals.

Usage

```
spearson(object)
```

Arguments

object	fitted model.

Details

Takes ordinary Pearson residuals and standardizes them.

Value

Returns the residuals.

Author(s)

Kristian Hovde Liland

Examples

```
data <- data.frame(y = rnorm(8),
   x = factor(c('a','a','a','a','b','b','b','b')),
   z = factor(c('a','a','b','b','a','a','b','b')))
mod <- lm(y ~ x + z, data=data)
spearson(mod)
```

tally *Tally of discrete numbers*

Description

Tally of discrete numbers

Usage

```
tally(x)
```

Arguments

x	Discrete number vector.

Value

Returns the tally.

Author(s)

Kristian Hovde Liland

Examples

```
tally(c(1,5,1,3,2,5,6,2,2,1,4,3,6))
```

t_test *Text book versions of t-tests and z-tests.*

Description

Adaptations of base t.test to better confrom to text book standards. t_test_sum and z_test_sum takes summarized data as input.

Usage

```
t_test(x, ...)
z_test(x, ...)

## Default S3 method:
t_test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, ...)
## Default S3 method:
z_test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.95, sds = NULL, ...)
```

```
## S3 method for class 'formula'
t_test(formula, data, subset, na.action, ...)
## S3 method for class 'formula'
z_test(formula, data, subset, na.action, ...)

## Function for summarized data:
t_test_sum(means, sds, ns, alternative = c("two.sided", "less", "greater"),
mu = 0, var.equal = FALSE, conf.level = 0.95, z.test = FALSE, ...)
z_test_sum(means, sds, ns, alternative = c("two.sided", "less", "greater"),
mu = 0, var.equal = FALSE, conf.level = 0.95, z.test = TRUE, ...)
```

Arguments

x	a (non-empty) numeric vector of data values.
y	an optional (non-empty) numeric vector of data values.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
mu	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
paired	a logical indicating whether you want a paired t-test.
var.equal	a logical variable indicating whether to treat the two variances as being equal. If TRUE then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
conf.level	confidence level of the interval.
formula	a formula of the form lhs ~ rhs where lhs is a numeric variable giving the data values and rhs a factor with two levels giving the corresponding groups.
data	an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
means	means of groups.
sds	standard deviations of groups.
ns	number of objects in groups.
z.test	normal approximation.
...	further arguments to be passed to or from methods.

Details

The formula interface is only applicable for the 2-sample tests.

alternative = "greater" is the alternative that x has a larger mean than y.

If paired is TRUE then both x and y must be specified and they must be the same length. Missing values are silently removed (in pairs if paired is TRUE). If var.equal is TRUE then the pooled estimate of the variance is used. By default, if var.equal is FALSE then the variance is estimated separately for both groups and the Welch modification to the degrees of freedom is used.

If the input data are effectively constant (compared to the larger of the two means) an error is generated.

Value

A list with class "htest" containing the following components:

statistic	the value of the t-statistic.
parameter	the degrees of freedom for the t-statistic.
p.value	the p-value for the test.
conf.int	a confidence interval for the mean appropriate to the specified alternative hypothesis.
estimate	the estimated mean or difference in means depending on whether it was a one-sample test or a two-sample test.
null.value	the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
alternative	a character string describing the alternative hypothesis.
method	a character string indicating what type of t-test was performed.
data.name	a character string giving the name(s) of the data.

See Also

prop.test

Examples

```
t.test(1:10,y=c(7:20))      # P = .00001855
t.test(1:10,y=c(7:20, 200)) # P = .1245    -- NOT significant anymore

## Classical example: Student's sleep data
plot(extra ~ group, data = sleep)
## Traditional interface
with(sleep, t.test(extra[group == 1], extra[group == 2]))
## Formula interface
t_test(extra ~ group, data = sleep)
```

Index