Package 'complexlm'

August 8, 2023

Version 1.1

Date 2023-08-08

Depends R (>= 3.5.0), MASS

Imports grDevices, pracma, graphics, stats, mathjaxr

Suggests dplyr, ggforce, ggplot2, reshape2, methods, utils

Description Tools for linear fitting with complex variables. Includes ordinary least-squares (zlm()) and robust M-estimation (rzlm()), and complex methods for oft used generics. Originally adapted from the rlm() functions of 'MASS' and the lm() functions of 'stats'.

Title Linear Fitting for Complex Valued Data

LazyData yes

ByteCompile yes

License GPL-2 | GPL-3

URL https://github.com/QuantumOfMoose/complexlm

Contact <wryan1@binghamton.edu>

NeedsCompilation yes

Author William Ryan [aut, cre, cph],
Bruce White [ths],
Brian Ripley [ctb, cph] (Wrote 'MASS' rlm() code upon which complexlm rlm() is based),
Bill Venables [ctb] (Wrote 'MASS' rlm() code upon which complexlm rlm() is based),
John Maindonald [ctb] (Wrote plot.lm() on which plot.zlm() is based),
Martin Maechler [ctb] (Wrote plot.lm() on which plot.zlm() is based),
R Core Team [ctb, cph] (Several functions in 'complexlm' are based on similar functions in 'base' or 'stats')

Maintainer William Ryan <wryan1@binghamton.edu>

Repository github

Date/Publication 2021-05-03 09:03:50 UTC

Roxygen list(markdown = TRUE)

RdMacros mathjaxr

RoxygenNote 7.1.2

Encoding UTF-8

R topics documented:

`anova.zlm` *ANOVA for Complex Linear Fits*

Description

A very simple adaptation of stats::anova.lm which can handle fits of complex variables. The only change was to take the absolute value of squared residuals, and eliminate quantile based features. Note that this function uses the variance, not the pseudo-variance. An analysis of pseudo-variance (ANOPVA) is also possible (and maybe useful), but not yet implemented.

Usage

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

## S3 method for class 'zlmlist'
anova(object, ..., scale = 0, test = "F")
```

Arguments

`object`	objects of class "zlm", usually produced by lm.
`...`	Other arguments.
`scale`	numeric. An estimate of the noise variance σ^2. If zero this will be estimated from the largest model considered.
`test`	a character string specifying the test statistic to be used. Can be one of `"F"`, `"Chisq"` or `"Cp"`, with partial matching allowed, or `NULL` for no test.

Details

Specifying a single object gives a sequential analysis of variance table for that fit. That is, the reductions in the residual sum of squares as each term of the formula is added in turn are given in as the rows of a table, plus the residual sum of squares.

The table will contain F statistics (and P values) comparing the mean square for the row to the residual mean square.

If more than one object is specified, the table has a row for the residual degrees of freedom and sum of squares for each model. For all but the first model, the change in degrees of freedom and sum of squares is also given. (This only make statistical sense if the models are nested.) It is conventional to list the models from smallest to largest, but this is up to the user.

Optionally the table can include test statistics. Normally the F statistic is most appropriate, which compares the mean square for a row to the residual sum of squares for the largest model considered. If `scale` is specified chi-squared tests can be used. Mallows' C_p statistic is the residual sum of squares plus twice the estimate of σ^2 times the residual degrees of freedom.

Value

An object of class "anova", which inherits from class "data.frame". Contains a analysis of variance table, except for those components that rely on quantiles.

Functions

- `anova.zlmlist`: s3 method for class 'zlmlist'

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.

See Also

lm, anova

Examples

```
set.seed(4242)
n <- 8
slop <- complex(real = 4.23, imaginary = 2.323)
interc <- complex(real = 1.4, imaginary = 1.804)
```

```
err <- complex(real = rnorm(n)/16, imaginary = rnorm(n)/16)
tframe <- data.frame(x= x <- complex(real=rnorm(n), imaginary= rnorm(n)), y=slop*x + interc+err)
fit <- lm(y ~ x, data = tframe, weights = rep(1,n))
anova(fit)
robfit <- rlm(y ~ x, data = tframe)
anova(fit, robfit)
```

`complexdqlrs` *An alternative to* .Call(C_Cdqlrs, x * wts, y * wts, tol, FALSE)) *that is compatible with complex variables*

Description

This serves as a wrapper for qr, replicating the behavior and output of the C++ function `C_Cdqlrs`. It is used in `zlm.wfit`, and is unlikely to be needed by end users.

Usage

```
complexdqlrs(x, y, tol = 1e-07, chk)
```

Arguments

`x`	a complex matrix (will also accept numeric, but in that case you might as well use `C_Cdqlrs`) whose QR decomposition is to be computed.
`y`	a complex vector or matrix of right-hand side quantities.
`tol`	the tolerance for detecting linear dependencies in the columns of x. Not used for complex `x`.
`chk`	not used. Included to better imitate `C_Cdqlrs`.

Value

A list that includes the qr decomposition, its coeffcionts, residuals, effects, rank, pivot information, qraux vector, tolerance, and whether or not it was pivoted. This is the same output as `C_Cdqlrs`.

See Also

base::qr, lm, zlm.wfit, rlm

Examples

```
set.seed(4242)
n <- 8
slope <- complex(real = 4.23, imaginary = 2.323)
intercept <- complex(real = 1.4, imaginary = 1.804)
x <- complex(real = rnorm(n), imaginary = rnorm(n))
y <- slope * x + intercept
complexdqlrs(x, y, 10^-4, 1.2)
```

`cooks.distance.zlm` *Cook's Distance for Complex Linear Models*

Description

Calculates the Cook's distances (technically a divergence, i.e. distance squared) of a complex linear model. These serve as a measurement of how much each input data point had on the model.

Usage

```
## S3 method for class 'zlm'
cooks.distance(model, lever = zhatvalues(model), ...)
```

Arguments

`model`	An object of class "lm" or "rlm". Can be complex or numeric.
`lever`	A list of leverage scores with the same length as `model$residuals`. By default zhatvalues is called on `model`.
`...`	Other parameters. Only used if `model` is numeric; in which case they are passed to `stats::cooks.distance`.

Details

Consider a linear model relating a response vector `y` to a predictor vector `x`, both of length `n`. Using the model and predictor vector we can calculate a vector of predicted values `yh`. `y` and `yh` are points in a `n` dimensional output space. If we drop the `i`-th element of `x` and `y`, then fit another model using the "dropped `i`" vectors, we can get another point in output space, `yhi`. The squared Euclidean distance between `yh` and `yhi`, divided by the rank of the model (`p`) times its mean squared error `s^2`, is the `i`-th Cook's distance.

$$D_i = (yh - yhi)^t (yh - yhi) / ps^2$$

A more elegant way to calculate it, which this function uses, is with the influence scores, `hii`.

$$D_i = |r_i|^2 / ps^2 hii / (1 - hii)$$

Where r_i is the i-th residual, and ^{t} is the conjugate transpose.

Value

A numeric vector. The elements are the Cook's distances of each data point in `model`.

Note

This is a simpler function than stats::cooks.distance, and does not understand any additional parameters not listed in this entry.

References

R. D. Cook, Influential Observations in Linear Regression, Journal of the American Statistical Association 74, 169 (1979).

See Also

stats::cooks.distance, zhatvalues

Examples

```
set.seed(4242)
n <- 8
slop <- complex(real = 4.23, imaginary = 2.323)
interc <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
xx <- complex(real= rnorm(n), imaginary= rnorm(n))
tframe <- data.frame(x = xx, y= slop*xx + interc + e)
fit <- lm(y ~ x, data = tframe, weights = rep(1,n))
cooks.distance(fit)
```

---

cov *Variance, Covariance, and Correlation for Complex Data*

Description

Wrappers of stats::var, stats::cov, and stats::cor that are capable of handling complex input.

Usage

```
cov(
  x,
  y = NULL,
  na.rm = FALSE,
  method = "pearson",
  use = "everything",
  pseudo = FALSE,
  ...
)

cor(
  x,
  y = NULL,
  na.rm = FALSE,
  use = "everything",
  method = "pearson",
  pseudo = FALSE,
  ...
)
```

```
var(x, y = NULL, na.rm = FALSE, use = "everything", pseudo = FALSE, ...)
```

Arguments

x	a numeric or complex vector, matrix, or dataframe.
y	NULL (default) or a numeric vector, matrix, or dataframe with dimensions compatible with x.
na.rm	logical. Should missing values be removed? Only considered when x is a vector.
method	The method for calculating correlation coefficient. Only "pearson" is supported for complex variables, so this parameter is ignored.
use	character string giving the desired method of computing covariances in the presence of missing values. Options are "everything" (default), "all.obs", "complete.obs", or "na.or.complete". See stats::cov for explanation of what each one does. Note that "pairwise.complete.obs" is not available for this complex method.
pseudo	logical, if TRUE the pseudo variance, covariance, or correlation is calculated. i.e. no complex conjugation is performed.
...	Other parameters, ignored.

Details

For vector input, the sample variance is calculated as,
$sum(Conj(mean(x) - x) * (mean(x) - x))/(length(x) - 1)$
And the sample covariance is calculated as,
$sum(Conj(mean(x) - x) * (mean(y) - y))/(length(x) - 1)$
The Pearson correlation coefficient, which is the only kind available for complex data, is the covariance divided by the product of the standard deviations of all variables. If `pseudo = TRUE`, these same expressions, sans `Conj()`, are used to calculate the pseudo, AKA relational, versions of variance, covariance, or correlation.

Value

numeric or complex the sample variance, covariance, or correlation of the input data.

Functions

- `cor`: Correlation coefficient of complex variables.
- `var`: S3 Variance or Pseudo Variance of Complex Variables, a synonym for cov.

Examples

```
set.seed(4242)
n <- 9
foo <- complex(real = rnorm(n), imaginary = rnorm(n))
var(foo)
bar <- complex(real = rnorm(n), imaginary = rnorm(n))
var(x = foo, y = bar)
```

```
foobar <- data.frame(foo, bar)
cov(foobar)
cor(foobar)
```

`CuHallData` *AC (Complex) Hall effect data measured from a thin-film copper sample.*

Description

This data serves as a 'real-world' example of complex valued measurements that can be analyzed by the methods contained in 'complexlm'. The Hall effect is a perpendicular (to the magnetic field and current) voltage that appears across an electrically conductive material when it is placed in an external magnetic field. It is commonly used to determine the mobility and density of charge carriers (electrons and/or holes) within materials.

Usage

```
CuHallData
```

Format

A dataframe with 240 rows and 11 columns. Most names contain the units of the column after the last or second to last period.

Details

The AC Hall effect is a more advanced technique that uses a time varying (sinusoidal) magnetic field and lock-in voltage measurement. The measured output signal is thus a wave, and best described by complex numbers. This strategy drastically lowers the noise floor, enabling measurement of difficult, low-mobility, materials.

This data was take at room temperature in a vacuum chamber using a custom built and programmed system. A Keithley 2636 sourcemeter provided the excitation current, while an SRS 830 lock-in amplifier measured the voltage signal from the sample. The sample consisted of a 0.8 mm square of 1000 Angstroms thick evaporated copper film on a 1.5 centimeter square die of oxidized crystalline silicon.

The variables included in the dataframe are as follows:

- Temperature.K.The temperature of the sample during this measurement. Units of Kelvin.
- Magnet.Field.Frequency.Hz.The frequency of the oscillating magnetic field used for this measurement. Units are Hertz.
- Magnetic.Field.T.The amplitude of the magnetic field during this measurement. Units of Tesla.
- Contact.ArrangementThis measurement involves four electrical contacts (current in and out, and voltage hi and lo), placed at the corners. Corresponding contacts must be opposite each other, so there are two possible arrangements: "D" and "F".

- Current.Set.A.The desired current to be sent through the sample for this measurement. There was an error in recording beyond the 8th row. Units of Amperes.
- Current.In.meas.A.The current measured leaving the sourcemeter, towards the sample. Units of Amperes.
- Current.Out.meas.A.The current measured exiting the sample. Units of Amperes.
- Source.V.V.The voltage generated across the sample in order to produce the desired current. Units of Volts.
- OutputVThe complex voltage measured across the sample. Ideally proportional to the current and magnetic field. Units of Volts.
- Current.leakThe difference between the input and output currents. Units of Amperes.
- Resistivity.Ohm.cmThe resistivity of the sample as calculated by the Van Der Pauw method from previous DC current-voltage sweeps. Units of Ohm*centimeter.
- thickness.cm.The thickness of the copper film in centimeters.

Author(s)

William Ryan <wryan1@binghamton.edu>

`lm` *Linear Model Fitting for Complex or Numeric Variables*

Description

An adaptation of lm that is compatible with complex variables. If the response is not complex, it calls the standard `stats::lm()` Note: It is not capable of dealing with contrasts in the complex case. The formula interpretation is also unable of handle algebraic expressions in `formula`.

Usage

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

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. For complex variables there are restrictions on what kinds of formula can be comprehended. See zmodel.matrix for 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	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
x	logical. If TRUE return the model matrix of the fit. Default is TRUE.
y	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
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	Not implemented for complex variables. See zmodel.matrix for details. Default is NULL
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 or matrix of extents matching those of the response. 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.
...	additional arguments to be passed to the low level regression fitting functions (see below).

Details

Like stats::lm the models are specified symbolically using the standard R formula notation: response ~ terms Meaning response is a linear combination of the predictor variables in terms. For compatibility with complex numbers complexlm::lm uses zmodel.matrix to build the model matrix

from the model formula. As such it is limited to terms consisting of predictor names connected by + and/or : operators. Anything more complicated will result in an error.

If response is a matrix, then then lm() fits a model to each column, and returns an object with class "mlm".

For complex input, this function calls zlm.wfit.

Value

Returns an object of class c("zlm","lm"), or for multiple responses c("zlm","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 functions coefficients, effects, fitted.values and residuals extract various useful features of the value returned by lm. Of course these things can also be accessed by simply using the get element symbol $.

Objects of class "zlm" are lists with the following components.

coefficients	a named vector of coefficients.
residuals	the residuals, that is response minus fitted values.
fitted.values	The fitted values, which are response values obtained by feeding the predictors into the model.
rank	The numeric rank of the fitted linear model.
weights	Numeric. The user supplied weights for the linear fit. If none were given, a vector of 1s of length equal to that of the input data.
df.residual	The residual degrees of freedom.
call	The matched call.
terms	the terms object used.
contrasts	The contrasts used, as these are not supported this component will probably be NULL.
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	the model matrix used, unless requested to not return it.
model	if requested, the model frame used.
na.action	(where relevant) information returned by model.frame on the special handling of NAs.
assign	Used by extractor functions like summary and effects to understand variable names. Not included in null fits.
effects	Complex list. See effects for explanation. Not included in null fits.
qr	unless declined, the output of the qr object created in the process of fitting. Not included in null fits.

Note

In complexlm, the x parameter defaults to TRUE so that followup methods such as predict.lm have access to the model matrix.

See Also

lm.fit, lm.wfit, zlm.wfit, zmodel.matrix, complexdqlrs, stats::lm

Examples

```
set.seed(4242)
n <- 8
slop <- complex(real = 4.23, imaginary = 2.323)
interc <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
xx <- complex(real= rnorm(n), imaginary= rnorm(n))
tframe <- data.frame(x= xx, y= slop*xx + interc + e)
lm(y ~ x, data = tframe, weights = rep(1,n))
```

`lm.fit` *Complex Variable Compatible Wrappers for* `stats::lm.fit()` *and* `stats::lm.wfit()`

Description

This function is just an if statement. If the design matrix `x` is complex, `zlm.wfit()` is called. Otherwise `stats::lm.fit()` or `stats::lm.wfit()` is called. These functions are unlikely to be needed by end users, as they are called by `lm()`.

Usage

```
lm.fit(
  x,
  y,
  offset = NULL,
  method = "qr",
  tol = 1e-07,
  singular.ok = TRUE,
  ...
)

lm.wfit(
  x,
  y,
  w,
  offset = NULL,
  method = "qr",
  tol = 1e-07,
  singular.ok = TRUE,
  ...
)
```

Arguments

x	design matrix of dimension n * p.
y	vector of observations of length n, or a matrix with n rows.
offset	(numeric of length n). This can be used to specify an *a priori* known component to be included in the linear predictor during fitting.
method	currently, only method = "qr" is supported.
tol	tolerance for the qr decomposition. Default is 1e-7.
singular.ok	logical. If FALSE, a singular model is an error.
...	currently disregarded.
w	vector of weights (length n) to be used in the fitting process for the wfit functions. Weighted least squares is used with weights w, i.e., sum(w * e^2) is minimized.

Value

a list with components (for lm.fit and lm.wfit)

coefficients	p vector
residuals	n vector or matrix
fitted.values	n vector or matrix
effects	n vector of orthogonal single-df effects. The first rank of them correspond to non-aliased coefficients, and are named accordingly.
weights	n vector — *only* for the *wfit* functions.
rank	integer, giving the rank
df.residual	degrees of freedom of residuals
qr	the QR decomposition, see qr.

Fits without any columns or non-zero weights do not have the effects and qr components.

.lm.fit() returns a subset of the above, the qr part unwrapped, plus a logical component pivoted indicating if the underlying QR algorithm did pivot.

Functions

- lm.wfit: wrapper for weighted linear fitting function.

Examples

```
set.seed(4242)
n <- 6
p <- 2
slop <- complex(real = 4.23, imaginary = 2.323)
slop2 = complex(real = 2.1, imaginary = -3.9)
interc <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
desmat <- matrix(c(complex(real = rnorm(n * p), imaginary = rnorm(n * p)), rep(1, n)), n, p + 1)
y = desmat %*% c(slop, slop2, interc) + e
lm.fit(desmat, y)
```

mad *Median Absolute Deviation, compatible with complex variables*

Description

Median absolute deviation, adapted to operate on complex data as well as numeric. In the later case it simply calls `stats::mad()`. For complex x it uses the geometric median, `pracma::geo_median()`, as the center, then returns the median absolute difference between center and each element of x, multiplied by `constant`.

Usage

```
mad(
  x,
  center = median(x),
  constant = 1.4826,
  na.rm = FALSE,
  low = FALSE,
  high = FALSE
)
```

Arguments

`x`	a numeric or complex vector.
`center`	optional, numeric or complex. The center about which to calculate MAD. Defaults to median for numeric, and geo_median for complex.
`constant`	a constant by which to multiply the median absolute deviation from center. Default is 1.4826, which is the inverse of the 3/4 quantile for the normal distribution.
`na.rm`	logical. Should NAs be removed from x before calculating.
`low`	logical. If TRUE, compute the "lo-median", i.e., for even sample size, do not average the two middle values, but take the smaller one. Not used if x is complex.
`high`	logical. If TRUE, compute the "hi-median", i.e., take the larger of the two middle values for even sample size. Not used if x is complex.

Value

numeric. The median absolute deviation (MAD) from center.

Note

The concept of quantile requires ordering to be defined, which the complex numbers lack. The usefulness of multiplying by `constant` is thus called into question. However, for no more rigorous reason than consistency, the default behavior of this function is to do so.

See Also

median.complex, stats::mad

Examples

```
set.seed(4242)
n <- 8
foo <- complex(real = rnorm(n), imaginary = rnorm(n))
mad(foo)
```

`mahalanobis` *Mahalanobis Distance, with better complex behavior*

Description

The Mahalanobis distance function included in the `stats` package returns a complex number when given complex values of `x`. But a distance (and thus its square) is always positive real. This function calculates the Mahalanobis distance using the conjugate transpose if given complex data, otherwise it calls stats::mahalanobis.

Usage

```
mahalanobis(x, center, cov, pcov = NULL, inverted = FALSE, ...)
```

Arguments

`x`	A length p vector or matrix with row length p. Or, a length $2p$ vector or matrix with row length $2p$.
`center`	A vector of length equal to that of `x`.
`cov`	The covariance matrix (pxp) of the distribution. Or, the "double covariance matrix" of the distribution, which contains the information from `cov` and `pcov` in a single $(2px2p)$ matrix. Can be generated by matrixweave, vcov.zlm, or vcov.rzlm. vcov.rzlm].
`pcov`	The pseudo covariance matrix (pxp) of the distribution. Optional.
`inverted`	Boolean, if TRUE, `cov` and `pcov` are not taken to be the *inverse* covariance and pseudo covariance matrices.
`...`	Optional arguments to be passed to solve, which is used for computing the inverse of `cov`. If `inverted = TRUE`, unused.

Details

Depending on the relative sizes of `x`, `cov`, and `pcov`, the function will perform slightly different calculations. If `pcov` is not included, the Mahalanobis distance is calculated using only `cov`. In this case if the dimension of `cov` is twice that of `x`, `x` is interleaved with its complex conjugate so that it becomes the same length as `cov`. Note that in this case the resulting Mahalanobis distance will only incorporate information about the interactions between the real and imaginary components if

the "double covariance matrix is given as cov . If pcov is included in the input, pcov and cov are interleaved to form the "double covariance", and this is used to calculate the Mahalanobis distance, interleaving x if necessary. This gives the user a great deal of flexibility when it comes to input.

Value

numeric. The squared Mahalanobis distance (divergence) between x and center.

References

D. Dai and Y. Liang, High-Dimensional Mahalanobis Distances of Complex Random Vectors, Mathematics 9, 1877 (2021).

See Also

matrixweave

Examples

```
set.seed(4242)
n <- 8
x <- matrix(complex(real = rnorm(n), imaginary = rnorm(n)), ncol = 2)
mu <- complex(real = 1.4, imaginary = 0.4)
sigma <- 3.4
mahalanobis(x, mu, sigma * diag(2))
```

---

matrixweave *Combine covariance matrix and pseudo covariance matrix into a "double covariance matrix"*

Description

Interleaves the elements of a (pxp) matrix with those of a different (pxp) matrix to form a $(2px2p)$ matrix. This function was originally made to combine the covariance and pseudo covariance matrices of a complex random vector into a single "double covariance matrix", as described in (van den Bos 1995). However, it will accept and operate on matrices of any storage mode.

Usage

```
matrixweave(cov, pcov, FUNC = Conj)
```

Arguments

cov	A square matrix, such as one describing the covariance between two complex random vectors.
pcov	A square matrix with the same size as cov. Perhaps a pseudo covariance matrix.
FUNC	A function to operate on the elements of pcov. The results of which will be a quarter of the elements of the returned matrix. Default is Conj.

Value

A square matrix with dimension twice that of the input matrices. Each element of which is an element from one of the inputs, and its nearest non-diagonal neighbors are from the other input. Half of the elements from pcov present in the output matrix are replaced by FUNC operated on them. Thus if two 2x2 matrices, A and B are given to `matrixweave()`, the elements of the result are:
matrixweave(A,B)[i,j] = if(i+j is even) A[ceiling(i/2), ceiling(j/2)]
if(i+j is odd and i > j) B[ceiling(i/2), ceiling(j/2)]
if(i+j is odd and i < j) FUNC(B[ceiling(i/2),ceiling(j/2)])

References

A. van den Bos, The Multivariate Complex Normal Distribution-a Generalization, IEEE Trans. Inform. Theory 41, 537 (1995).

See Also

mahalanobis, vcov.zlm, vcov.rzlm

Examples

```
set.seed(4242)
mata <- matrix(rnorm(9), nrow = 3)
matb <- matrix(rnorm(9), nrow = 3)
matrixweave(mata, matb)
```

`median.complex` *Improved Median for Complex Numbers*

Description

By default stats::median handles complex numbers by computing the medians of the real and imaginary components separately. This is not ideal as the result is not rotationally invariant (the same set of numbers will have a different median if the coordinates are rotated). This method calculates the complex median as the geometric median, as implemented in pracma::geo_median, which is rotationally invariant.

Usage

```
## S3 method for class 'complex'
median(x, na.rm = FALSE, tol = 1e-07, maxiter = 200, ...)
```

Arguments

x	a numeric or complex vector, of which the median is to be calculated.
na.rm	logical. Should NA values be removed before finding the median?
tol	numeric. Relative tolerance to be passed to pracma::geo_median. Default is 1e-07.

`maxiter`	maximum number of iterations for calculating geometric median. Not used if x is numeric.
`...`	Additional arguments. Currently ignored.

Details

The geometric median fails if any of the input data are colinear. Meaning that a straight line on the complex plane can be drawn which intersects with one or more element of `x`. If this function encounters such an error, it adds a small amount of random jitter to the data, then calculates the geometric medium again. The jitter is generated by a normal distribution with a standard deviation equal to the absolute minimum real or imaginary component of `x`, divided by 10^8 (or 10^-9 if the minimum is zero).

Value

The median of x. If x is complex, the geometric median as calculated by Weiszfeld's algorithm.

Note

This method masks the default method for complex numbers.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

stats::median and pracma::geo_median

Examples

```
set.seed(4242)
n <- 7
foo <- complex(real = rnorm(n), imaginary = rnorm(n))
median(foo)
```

---

plot.zlm *Plot Diagnostics for a Complex Linear Model Objects*

Description

A modified version of stats::plot.lm used for visualizing ordinary ("zlm") and robust ("rzlm") linear models of complex variables. This documentation entry describes the complex version, focusing on the differences and changes from the numeric. For further explanation of the plots please see stats::plot.lm.

Usage

```
## S3 method for class 'zlm'
plot(
  x,
  which = c(1, 3, 5),
  caption = list("Residuals vs Fitted", "Scale-Location", "Cook's distance",
   "Residuals vs Leverage", expression("Cook's dist vs Leverage  " * h[ii]/(1 - h[ii]))),
 panel = if (add.smooth) function(x, y, ...) panel.smooth(x, y, iter = iter.smooth,
    ...) else points,
  sub.caption = NULL,
  main = "",
  ask = prod(par("mfcol")) < length(which) && dev.interactive(),
  ...,
  id.n = 3,
  labels.id = names(residuals(x)),
  cex.id = 0.75,
  cook.levels = c(0.5, 1),
  add.smooth = getOption("add.smooth"),
  iter.smooth = if (isGlm) 0 else 3,
  label.pos = c(4, 2),
  cex.caption = 1,
  cex.oma.main = 1.25
)
```

Arguments

x	complex lm object ("zlm" or "rzlm"). Typically produced by lm or rlm.
which	If a subset of the plots is required, specify a subset of the numbers 1:6, except 2. See stats::plot.lm, and below, for the different kinds. Default is c(1,3,5).
caption	captions to appear above the plots; character vector or list of valid graphics annotations, see as.graphicsAnnot, of length 6, the j-th entry corresponding to which[j]. Can be set to "" or NA to suppress all captions.
panel	panel function. The useful alternative to points, panel.smooth can be chosen by add.smooth = TRUE.
sub.caption	common title—above the figures if there are more than one; used as sub (s.title) otherwise. If NULL, as by default, a possible abbreviated version of deparse(x$call) is used.
main	title to each plot—in addition to caption.
ask	logical; if TRUE, the user is *ask*ed before each plot, see par(ask=.).
...	other parameters to be passed through to plotting functions.
id.n	number of points to be labelled in each plot, starting with the most extreme.
labels.id	vector of labels, from which the labels for extreme points will be chosen. NULL uses observation numbers.
cex.id	magnification of point labels.
cook.levels	levels of Cook's distance at which to draw contours.

add.smooth	logical indicating if a smoother should be added to most plots; see also panel above.
iter.smooth	the number of robustness iterations, the argument iter in panel.smooth(); the default uses no such iterations for glm fits which is particularly desirable for the (predominant) case of binary observations, but also for other models where the response distribution can be highly skewed.
label.pos	positioning of labels, for the left half and right half of the graph respectively, for plots 1-3.
cex.caption	controls the size of caption.
cex.oma.main	controls the size of the sub.caption only if that is *above* the figures when there is more than one.

Details

Five of the six plots generated by stats::plot.lm can be produced by this function: The residuals vs. fitted values plot, the scale-location plot, the plot of Cook's distances vs. row labels, the plot of residuals vs. leverages, and the plot of Cook's distances vs. leverage/(1-leverage). The Q-Q plot is *not* drawn because it requires quantiles, which are not unambiguously defined for complex numbers. Because complex numbers are two dimensional, pairs is used to create multiple scatter plots of the real and imaginary components for the residuals vs. fitted values and scale-location plots.

Value

Several diagnostic plots.

References

Belsley, D. A., Kuh, E. and Welsch, R. E. (1980). Regression Diagnostics. New York: Wiley.

Cook, R. D. and Weisberg, S. (1982). Residuals and Influence in Regression. London: Chapman and Hall.

Firth, D. (1991) Generalized Linear Models. In Hinkley, D. V. and Reid, N. and Snell, E. J., eds: Pp. 55-82 in Statistical Theory and Modelling. In Honour of Sir David Cox, FRS. London: Chapman and Hall.

Hinkley, D. V. (1975). On power transformations to symmetry. Biometrika, 62, 101-111. doi: 10.2307/2334491.

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

See Also

zhatvalues, cooks.distance, lm, rlm

Examples

```
set.seed(4242)
n <- 8
slop <- complex(real = 4.23, imaginary = 2.323)
```

```
interc <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
xx <- complex(real= rnorm(n), imaginary= rnorm(n))
tframe <- data.frame(x = xx, y= slop*xx + interc + e)
fit <- lm(y ~ x, data = tframe, weights = rep(1,n))
plot(fit)
```

---

psi.huber *Weighting functions for robust fitting*

Description

Weighting functions used in rlm for iteratively reweighted least squares. Based on the same functions from MASS, modified to accept complex variables. While named 'psi', these are actually weight functions, weight(u) = abs(influence(u) / u).

Usage

```
psi.huber(u, k = 1.345, deriv = 0)

psi.hampel(u, a = 2, b = 4, c = 8, deriv = 0)

psi.bisquare(u, c = 4.685, deriv = 0)
```

Arguments

u	Numeric or complex. When used in M-estimation, a residual.
k	Numeric. A scaling constant for psi.huber. Default value is 1.345
deriv	Numeric. If 0, return the weight at u. If 1 return the first derivative of the influence function at u.
a	Numeric. A scaling constant for psi.hampel. Default value is 2
b	Numeric. A scaling constant for psi.hampel. Default value is 4
c	Numeric. A scaling constant for 'psi.hampelorpsi.bisquare'. Default is 8 for the former and 4.685 for the later.

Details

Three weight functions for iteratively (re)weighted least squares. When used in M-estimation, psi.huber has a unique solution. psi.hampel and psi.bisquare, which are redescending M-estimators, do not have a unique solution. These are capable of completely rejecting outliers, but need a good starting point to avoid falling into unhelpful local minima. If deriv is set to 1, the functions return the value of the first derivative of the influence function at u. Note that they do not return the derivative of the weight function, as might be expected.

Value

A numeric or complex that is either the value of the weight function at u (numeric) or the first derivative of the influence function at u (can be complex).

Functions

- `psi.hampel`: The weight function of the hampel objective function.
- `psi.bisquare`: The weight function of Tukey's bisquare objective function.

Examples

```
set.seed(4242)
z <- complex(real = rnorm(3), imaginary = rnorm(3))
psi.huber(z)
psi.hampel(z)
psi.bisquare(z)
psi.huber(z, deriv=1)
psi.hampel(z, deriv=1)
```

---

range *Range For Complex Objects*

Description

This function extends base::range to the field of complex numbers. It returns a vector containing two complex numbers that are the diagonal points of a rectangle, with sides parallel to the real and imaginary axes, that just contains all the complex numbers given as arguments. If given non complex input it calls base::range, please see the documentation for that function for an explanation of its behavior with other input.

Usage

```
range(..., na.rm = FALSE, finite = FALSE)
```

Arguments

`...`	Any complex, numeric, or character object
`na.rm`	logical, indicates if `NA`'s should be removed.
`finite`	logical, indicates if non-finite elements should be omitted.

Value

A complex vector describing a rectangle that all input values fall within.

See Also

base::range

Examples

```
set.seed(4242)
n <- 8
foo <- complex(real = rnorm(n), imaginary = rnorm(n))
range(foo)
```

rlm *Robust Fitting of Linear Models, Compatible with Complex Variables*

Description

Uses robust M-estimation to fit a linear model to numeric or complex data. Based on MASS::rlm.

Usage

```
rlm(x, ...)

## S3 method for class 'formula'
rlm(
  formula,
  data,
  weights,
  ...,
  subset,
  na.action,
  method = c("M", "MM", "model.frame"),
  wt.method = c("inv.var", "case"),
  model = TRUE,
  x.ret = TRUE,
  y.ret = FALSE,
  contrasts = NULL
)

## S3 method for class 'complex'
rlm(
  x,
  y,
  weights,
  ...,
  w = rep(1, nrow(x)),
  init = "ls",
  psi = psi.huber,
  scale.est = c("MAD", "Huber", "proposal 2"),
  k2 = 1.345,
  method = c("M", "MM"),
  wt.method = c("inv.var", "case"),
  maxit = 20,
  acc = 1e-04,
  test.vec = "resid",
  lqs.control = NULL,
  interc = FALSE
)
```

Arguments

x	numeric or complex. A matrix, dataframe, or vector containing the explanatory / independent / predictor variables.
...	additional arguments to be passed to rlm.default or to the psi function.
formula	a formula object of the form y ~ x1 + x2. Note that algebraic expressions in formula cannot currently be used with complex data.
data	a data frame containing the variables upon which a robust fit is to be applied.
weights	numeric. A vector of weights to apply to the residuals.
subset	an index vector specifying the cases (rows of data or x and y) to be used for fitting.
na.action	a function that specifies what to do if NAs are found in the fitting data. The default is to omit them via na.omit. Can also be changed by options (na.action =).
method	string. What method of robust estimation should be used. Options are "M", "MM", or "model.frame". The default is M-estimation. MM-estimation has a high breakdown point but is not compatible with complex variables or case weights. model.frame just constructs the model frame, and only works with the formula method.
wt.method	string, either "inv.var" or "case". Specifies whether the weights are case weights that give the relative importance of each observation (higher weight means more important) / case, or the inverse variances of the cases (higher weight means that observation is less variable / uncertain).
model	logical. Should the model frame be included in the returned object?
x.ret	logical. Should the model (design) matrix be included in the returned object?
y.ret	logical. Should the response be included in the returned object?
contrasts	optional contrast specifications: see stats::lm. Not compatible with complex data.
y	numeric or complex. A vector containing the dependent / response variables, the same length as x.
w	(optional) initial down-weighting for each case
init	(optional) initial values for the coefficients OR a method to find initial values OR the result of a fit with a coef component. Known methods are "ls" (the default) for an initial least-squares fit using weights w*weights, and "lts" for an unweighted least-trimmed squares fit with 200 samples.
psi	the psi function is specified by this argument. It must give (possibly by name) a function g(x, ..., deriv) that for deriv=0 returns psi(x)/x and for deriv=1 returns psi'(x). Tuning constants will be passed in via ...
scale.est	method of scale estimation: re-scaled MAD of the residuals (default) or Huber's proposal 2 (which can be selected by either "Huber" or "proposal 2"). Only MAD is implemented for complex variables.
k2	tuning constant used for Huber proposal 2 scale estimation.
maxit	maximum number of IWLS iterations.

acc	the accuracy for the stopping criterion.
test.vec	the stopping criterion is based on changes in this vector.
lqs.control	An optional list of control values for lqs, ignored.
interc	TRUE or FALSE, default is FALSE. Used with rlm.default when fitting complex valued data. If true, a y-intercept is calculated during fitting. Otherwise, the intercept is set to zero.

Details

M-estimation works by finding the model coefficients that minimize the sum of a function of the residuals. This function, called the objective function rho(), is a kind of statistical distance (AKA divergence), and a semimetric. As a semimetric it is a function of the measured value y and the modeled value Y (residual $r = y - Y$) which maps from the space of the data to the positive real numbers. Semimetrics can be defined for domains of any dimensionality, including the two dimensional complex numbers, and thus so can M-estimation. What's more, all the standard algebraic operations used in the itteratively (re)weighted least-squares M-estimator robust regression algorithm are defined over the set of complex numbers. While ordering is not defined for them, it is the output of rho(), a real number, that must be in M-estimation.

Value

An object of class `c("rzlm","zlm","rlm","lm")`, or for numeric data `c("rlm","lm")`.

Objects of class "rzlm" are lists with the same components as "zlm" objects, as well as,

df.residual	NA For "rzlm" objects the residual degrees of freedom are always set to NA in order to avoid estimation of residual scale by "zlm" or "lm" methods.
s	The robust scale estimate used.
w	The weights used in the IWLS process.
psi	The psi (itterative weighting) function with parameters substituted in.
conv	The value of the convergence criteria at each iteration.
converged	Did the IWLS process converge?
wresid	A 'working residual', the residuals of the last re-weighted least-squares. Weighted by weights if "inv.var" weights were used.

See MASS::rlm for a description of "rlm" objects.

Methods (by class)

- formula: S3 method for class 'formula'
- complex: Complex Default S3 method

References

P. J. Huber (1981) *Robust Statistics*. Wiley.

F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw and W. A. Stahel (1986) *Robust Statistics: The Approach based on Influence Functions*. Wiley.

A. Marazzi (1993) *Algorithms, Routines and S Functions for Robust Statistics*. Wadsworth & Brooks/Cole.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

Examples

```
set.seed(4242)
n <- 8
slope <- complex(real = 4.23, imaginary = 2.323)
interc <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
xx <- complex(real= rnorm(n), imaginary= rnorm(n))
tframe <- data.frame(x = xx, y= slope*xx + interc + e)
rlm(y ~ x, data = tframe, weights = rep(1,n))
set.seed(4242)
n <- 8
slope <- complex(real = 4.23, imaginary = 2.323)
intercept <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
x <- complex(real = rnorm(n), imaginary = rnorm(n))
y <- slope * x + intercept + e
rlm(x = x, y = y, weights = rep(1,n), interc = TRUE)
```

`rstandard.zlm` *Standardized Residuals from Ordinary or Robust Linear fits with Complex Variables*

Description

Generates a vector of residuals from the given complex linear model that are normalized to have unit variance. Similar to stats::rstandard, which this function calls if given numeric input.

Usage

```
## S3 method for class 'zlm'
rstandard(model, lever = zhatvalues(model), ...)
```

Arguments

`model`	An object of class "zlm", "rzlm", "lm", or "rlm". Can be complex or numeric.
`lever`	A list of leverage scores with the same length as `model$residuals`. By default zhatvalues is called on `model`.
`...`	Other parameters. Only used if `model` is numeric; in which case they are passed to `stats::rstandard`.

Details

The standardized residuals are calculated as,

$$r' = r/(s\sqrt{1 - lever})$$

Where r is the residual vector and s is the residual standard error for "zlm" objects or the robust scale estimate for "rzlm" objects.

Value

A complex vector of length equal to that of the residuals of `model`. Numeric for numeric input.

Note

This is a much simpler function than stats::rstandard. It cannot perform leave-one-out cross validation residuals, or anything else not mentioned here.

See Also

stats::rstandard, stats::rstandard.lm,

Examples

```
set.seed(4242)
n <- 8
slop <- complex(real = 4.23, imaginary = 2.323)
interc <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
xx <- complex(real= rnorm(n), imaginary= rnorm(n))
tframe <- data.frame(x = xx, y= slop*xx + interc + e)
fit <- lm(y ~ x, data = tframe, weights = rep(1,n))
rstandard(fit)
```

---

summary.complex *summary method for complex objects*

Description

The base summary method for complex objects only reports their length and that they are complex.. This improved method returns the mean, median, variance, and pseudo variance of the given complex object.

Usage

```
## S3 method for class 'complex'
summary(object, ..., digits)
```

Arguments

object	a complex vector or scalar.
...	additional arguments, not used.
digits	integer specifying the number of digits to include in the summary values.

Value

A complex vector containing in order the length, median, mean, variance, and pseudo variance of the object. The length element will be a positive integer, despite being in the complex mode.

Examples

```
set.seed(4242)
n <- 8
foo <- complex(real = rnorm(n), imaginary = rnorm(n))
summary(foo)
```

---

summary.rzlm *Summary Method for Robust Linear Models*

Description

Summary method for objects of class "rzlm", capable of processing complex variable fits. If the residuals in the passed object are numeric, this function just calls `MASS::summary.rlm()`.

Usage

```
## S3 method for class 'rzlm'
summary(object, method = c("XtX", "XtWX"), correlation = FALSE, ...)
```

Arguments

object	An object inheriting from the class 'rzlm'. That is, a fitted model that was generated by rlm.
method	Character string indicating if the IWLS weights should be used when calculating matrix cross-products. "XtX" does not include weights, "XtWX" does.
correlation	Logical. Should the correlations be computed and printed?
...	Other arguments, passed to or from other methods.

Details

This is a method for the generic `summary()` function. It is based on the function of the same name from MASS, but has been modified to accept complex numbers. In addition to the standard error of the coefficients, it calculates a "pseudo standard error" or "relational standard error" as the square root of the "pseudo-variance". This is a complex number that quantifies the covariance between the real and imaginary parts. It can also be thought of as the amount and direction of anisotropy in the (presumed complex normal) probability distribution in the complex plane. The argument of this number gives the direction of the semi-major axis.

Value

Returns a list containing the following elements. If the list is printed by the function (`print.summary.rlm` or invoked as a method by `summary`), a null value is returned instead.

`correlation` A numeric matrix. The computed correlation coefficient matrix for the coefficients in the model.

`pseudocorrelation` A complex matrix. The computed pseudo-correlation coefficient matrix for the coefficients in the model.

`cov.unscaled` The unscaled covariance matrix; i.e, a numeric matrix such that multiplying it by an estimate of the error variance produces an estimated covariance matrix for the coefficients.

`pcov.unscaled` The unscaled pseudo-covariance matrix; i.e, a complex matrix such that multiplying it by an estimate of the error pseudo-variance produces an estimated pseudo-covariance matrix for the coefficients.

`sigma` Numeric. The scale estimate returned by `rlm`, which was used to scale the residuals before passing them to the `psi` weight function.

`stddev` Numeric. A scale estimate used for the standard errors. Calculated from the working residuals, the `psi` weight function, and the derivative of the influence function (`psi.method(deriv = 1)`).

`pstddev` Complex. A scale estimate used for the 'pseudo' standard errors. Calculated from the working residuals, the `psi` weight function, and the derivative of the influence function (`psi.method(deriv = 1)`). See details above.

`df` The number of degrees of freedom for the model and for residuals.

`coefficients` A 4 column matrix that contains the model coefficients, their standard errors, their pseudo standard errors (see details above), and their t statistics.

`terms` The terms object used in fitting this model.

References

W. N. Venables and B. D. Ripley, Modern Applied Statistics with S, 4th ed (Springer, New York, 2002). P. J. Huber and E. Ronchetti, Robust Statistics, 2nd ed (Wiley, Hoboken, N.J, 2009).

Examples

```
set.seed(4242)
n <- 8
slope <- complex(real = 4.23, imaginary = 2.323)
intercept <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
x <- complex(real = rnorm(n), imaginary = rnorm(n))
y <- slope * x + intercept + e
robfit <- rlm(x = x, y = y, weights = rep(1,n), interc = TRUE)
summary(robfit)
```

summary.zlm *Summarize Complex Linear Model Fits.*

Description

Summary method for complex linear fits of class "zlm". Based off of, and very similar to stats::summary.lm. However it does not delve into quantiles or 'significance stars', and includes the 'pseudo variance'.

Usage

```
## S3 method for class 'zlm'
summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)

## S3 method for class 'summary.zlm'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  symbolic.cor = x$symbolic.cor,
  quantiles = FALSE,
  ...
)
```

Arguments

object	An object of class "zlm". Presumably returned by lm. May contain complex variables.
correlation	Logical. If TRUE, the correlation matrix of the estimated parameters is returned and printed.
symbolic.cor	Logical. If TRUE, print the correlations in a symbolic form (see stats::symnum) rather than as numbers. (This may not work.)
...	Further arguments passed to or from other methods.
x	a 'zlm' object or an 'zlm' summary object. Used for `print.summary.zlm`
digits	the number of digits to include in the printed report, default is three. Used for `print.summary.zlm`
quantiles	logical. Should the (inaccurate) quantiles of the residuals be printed? If `FALSE` summary.complex is applied to the residuals instead.

Details

See stats::summary.lm for more information. In addition to the information returned by `stats::summary.lm`, this complex variable compatible version also returns "pseudo standard error" or "relational standard error" which is the square root of the "pseudo-variance". This is a complex number that quantifies the covariance between the real and imaginary parts. Can also be thought of as the amount and direction of anisotropy of the (presumed complex normal) probability distribution of the residuals in the complex plane. The argument of this number gives the direction of the semi-major axis of

an iso-probability-density ellipse centered on the mean, while its modulus is the length of the semi-major axis. The variance, meanwhile, gives the area of this ellipse, divided by pi. Together they fully describe it.

Value

Returns an object of class "summary.zlm" and "summary.lm", that is a list containing the following elements.

`residuals`	Complex or numeric. The weighted residuals, that is the measured value minus the fitted value, scaled by the square root of the weights given in the call to lm.
`correlation`	A complex matrix with real diagonal elements. The computed correlation coefficient matrix for the coefficients in the model.
`pseudocorrelation`	A complex matrix. The computed pseudo-correlation coefficient matrix for the coefficients in the model.
`cov.unscaled`	The unscaled covariance matrix; i.e, a complex matrix with real diagonal elements such that multiplying it by an estimate of the error variance produces an estimated covariance matrix for the coefficients.
`pcov.unscaled`	The unscaled pseudo-covariance matrix; i.e, a complex matrix such that multiplying it by an estimate of the error pseudo-variance produces an estimated pseudo-covariance matrix for the coefficients.
`sigma`	Numeric. The square root of the estimated variance of the random error.
`psigma`	Complex. The square root of the estimated pseudo-variance of the random error. See details above.
`df`	The number of degrees of freedom for the model and for residuals. A 3 element vector (p, n-p, p*), the first being the number of non-aliased coefficients, the last being the total number of coefficients.
`coefficients`	A 5 column matrix that contains the model coefficients, their standard errors, their pseudo standard errors (see details above), their t statistics, and corresponding (two-sided) p-value. Aliased coefficients are omitted.
`aliased`	Named logical vector showing if the original coefficients are aliased.
`terms`	The terms object used in fitting this model.
`fstatistic`	(for models including non-intercept terms) a 3 element numeric vector with the value of the F-statistic with its numerator and denominator degrees of freedom.
`r.squared`	Numeric. The fraction of variance explained by the model.
`adj.r.squared`	the above R^2 statistic "adjusted", penalizing for higher p.
`symbolic.cor`	(only if `correlation` is true.) The value of the argument symbolic.cor.
`na.action`	from `object`, if present there.

Functions

- `print.summary.zlm`: S3 method for class 'summary.zlm'

Note

`print.summary.zlm` calls `print.summary.rzlm`

For complex fits the quantiles reported by this function are based on sorting the real parts of the residuals. They should not be considered reliable..

See Also

lm, rlm

Examples

```
set.seed(4242)
n <- 8
slop <- complex(real = 4.23, imaginary = 2.323)
interc <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
xx <- complex(real= rnorm(n), imaginary= rnorm(n))
tframe <- data.frame(x = xx, y= slop*xx + interc + e)
fit <- lm(y ~ x, data = tframe, weights = rep(1,n))
summ <- summary(fit)
print(summ)
```

---

vcov.rzlm *Calculate Variance-Covariance Matrix and Pseudo Variance-Covariance Matrix for a Complex Fitted Model Object*

Description

A version of stats::vcov that is compatible with complex linear models. In addition to the variance-covariance matrix, the pseudo variance-covariance matrix, which is a measure of the covariance between real and imaginary components, is returned as well. Can also return the "big covariance" matrix, which combines the information of the covariance matrix and the pseudo-covariance matrix, as described in (van den Bos 1995). While not as compact as two seperate smaller matrices, the big covariance matrix simplifies calculations such as the mahalanobis distance.

Usage

```
## S3 method for class 'rzlm'
vcov(object, merge = TRUE, ...)
```

Arguments

`object`	a fitted model object, typically. Sometimes also a summary() object of such a fitted model.
`merge`	logical. Should the covariance matrix and pseudo-covariance / relational matrix be merged into one matrix of twice the dimensions? Default is TRUE.
`...`	Additional parameters, not currently used for anything.

Value

If merge is false, a list containing both the numeric variance-covariance matrix, and the complex pseudo variance-covariance matrix. If merge is true, a large matrix (both dimensions twice the number of coefficients) containing both the variance-covariance matrix and the pseudo variance-covariance matrix, merged together.

References

A. van den Bos, The Multivariate Complex Normal Distribution-a Generalization, IEEE Trans. Inform. Theory 41, 537 (1995).

Examples

```
set.seed(4242)
n <- 8
slope <- complex(real = 4.23, imaginary = 2.323)
intercept <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
x <- complex(real = rnorm(n), imaginary = rnorm(n))
y <- slope * x + intercept + e
robfit <- rlm(x = x, y = y, weights = rep(1,n), interc = TRUE)
summary(robfit)$stddev
vcov(robfit)
```

---

vcov.zlm *Calculate Variance-Covariance Matrix and Pseudo Variance-Covariance Matrix for a Complex Fitted Model Object*

Description

A method for of stats::vcov that is compatible with complex linear models. In addition to variance-covariance matrix, the pseudo variance-covariance matrix, which is a measure of the covariance between real and imaginary components, is returned as well. Can also return the "double covariance" matrix, which combines the information of the covariance matrix and the pseudo-covariance matrix, as described in (van den Bos 1995). While not as compact as two separate smaller matrices, the double covariance matrix simplifies calculations such as the mahalanobis distance.

Usage

```
## S3 method for class 'zlm'
vcov(object, complete = TRUE, merge = TRUE, ...)

.vcov.aliased.complex(aliased, vc, complete = TRUE)
```

Arguments

`object`	Typically a fitted model object of class "zlm" and/or "rzlm". Sometimes also a summary() object of such a fitted model.
`complete`	logical. Indicates if the full covariance and pseudo-covariance matrices should be returned even in the case of an over-determined system, meaning that some coefficients are undefined.
`merge`	logical. Should the covariance matrix and pseudo-covariance / relational matrix be merged into one matrix of twice the dimensions? Default is TRUE.
`...`	Additional parameters, not currently used for anything.
`aliased`	a logical vector typically identical to `is.na(coef(.))` indicating which coefficients are 'aliased'.
`vc`	a variance-covariance matrix, typically "incomplete", i.e., with no rows and columns for aliased coefficients.

Value

If `merge` is false, a list containing both the numeric variance-covariance matrix, and the complex pseudo variance-covariance matrix. If `merge` is true, a large matrix (both dimensions being twice the number of coefficients) containing both the variance-covariance matrix and the pseudo variance-covariance matrix, merged together.

Functions

- `.vcov.aliased.complex`: auxiliary function for dealing with singular model fits. See stats::vcov.

References

A. van den Bos, The Multivariate Complex Normal Distribution-a Generalization, IEEE Trans. Inform. Theory 41, 537 (1995).

Examples

```
set.seed(4242)
n <- 8
slop <- complex(real = 4.23, imaginary = 2.323)
interc <- complex(real = 1.4, imaginary = 1.804)
err <- complex(real = rnorm(n)/16, imaginary = rnorm(n)/16)
tframe <- data.frame(x= x <- complex(real=rnorm(n), imaginary= rnorm(n)), y=slop*x + interc+err)
fit <- lm(y ~ x, data = tframe, weights = rep(1,n))
vcov(fit)
```

`wmedian` *Weighted Median*

Description

This calculates the weighted median of a vector `x` using the weights in `w`. Weights are re-scaled based on their sum.

Usage

```
wmedian(x, w = rep(1, length(x)))
```

Arguments

`x`	numeric, a vector containing the data from which to calculate the weighted median.
`w`	numeric, a vector of weights to give the data in `x`.

Details

Sorts `x` and `w` by size of the elements of `x`. Then re-scales the elements of `w` to be between 0 and 1. Then sets n equal to the sum of all scaled weights with values less than 0.5. If the (n+1)-th element of the rescaled weights is greater than 0.5, the weighted median is the (n+1)-th element of the sorted `x`. Otherwise it is the average of the (n+1)-th and (n+2)-th elements of the sorted `x`.

Value

numeric. The weighted median of `x` using `w` as the weights.

Note

This is not compatible with complex data.

References

F. Y. Edgeworth, XXII. On a New Method of Reducing Observations Relating to Several Quantities, (1888). Also see the Wikipedia article on weighted median for a very good explanation and a model algorithm.

See Also

stats::median

Examples

```
xx <- rnorm(10, 4L, 1.5)
ww <- runif(10)
wmedian(xx, ww)
```

`zhatvalues` *Generate the Hat Matrix or Leverage Scores of a Complex Linear Model*

Description

This function returns either the full hat matrix (AKA the projection matrix) of a complex "lm" or "rlm" object, or the diagonal elements of same. The later are also known as the influence scores. It performs the same basic role as stats::hat and stats::hatvalues do for numeric fits, but is quite a bit simpler and rather less versatile.

Usage

```
zhatvalues(model, full = FALSE, ...)
```

Arguments

`model`	A complex linear fit object, of class "zlm" or "rzlm". An object with numeric residuals will produce a warning and NULL output.
`full`	Logical. If TRUE, return the entire hat matrix. If FALSE, return a vector of the diagonal elements of the hat matrix. These are the influence scores. Default is FALSE.
`...`	Additional arguments. Not used.

Details

For unweighted least-squares fits the hat matrix is calculated from the model matrix, X =`model$x`, as

$$H = X(X^t X)^{-1} X^t$$

For rlm or weighted least-squares fits the hat matrix is calculated as

$$H = X(X^t W X)^{-1} X^t W$$

Where ^{t} represents conjugate transpose, and W is the identity matrix times the user provided weights and the final IWLS weights if present.
Note that this function requires that the model matrix be returned when calling lm or rlm.
The diagonals will be purely real, and are converted to numeric if `full == FALSE`.

Value

Either a (nxn) complex matrix or a length n numeric vector.

See Also

stats::hatvalues, stats::hat, cooks.distance

Examples

```
set.seed(4242)
n <- 8
slop <- complex(real = 4.23, imaginary = 2.323)
interc <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
xx <- complex(real= rnorm(n), imaginary= rnorm(n))
tframe <- data.frame(x = xx, y= slop*xx + interc + e)
fit <- lm(y ~ x, data = tframe, weights = rep(1,n))
zhatvalues(fit)
```

---

zlm.wfit *Least-Squares Linear Fitting for Complex Variables*

Description

The function eventually called by `lm()`, `lm.fit()`, and/or `lm.wfit()` if fed complex data. Performs ordinary (least-squares) linear fitting on complex variable data. Like `stats::lm.wfit()`, which it is based off of, it uses qr decomposition for the matrix algebra. Unlike `stats::lm.wfit()` it also handles un-weighted regression, by setting the weights to 1 by default.

Usage

```
zlm.wfit(
  x,
  y,
  w = rep(1L, ifelse(is.vector(x), length(x), nrow(x))),
  offset = NULL,
  method = "qr",
  tol = 1e-07,
  singular.ok = TRUE,
  ...
)
```

Arguments

`x`	a complex design matrix, `n` rows by `p` columns.
`y`	a vector of observations/responses of length `n`, or a matrix with `n` rows.
`w`	a vector of weights to be used in the fitting process. The sum of `w * r^2` is minimized, with `r` being the residuals. By default, `w` is a vector of length `n` with every element equal to 1, making this an unweighted fit.
`offset`	optional. A complex vector of length `n` that will be subtracted from `y` prior to fitting.
`method`	optional. a string that can be used to choose any method you would like. As long as it is "qr".
`tol`	tolerance for the qr decomposition. Default is 1e-7.

singular.ok	logical. If false, a singular model is an error.
...	currently disregarded.

Value

a list with components (for lm.fit and lm.wfit)

coefficients	p vector
residuals	n vector or matrix
fitted.values	n vector or matrix
effects	n vector of orthogonal single-df effects. The first rank of them correspond to non-aliased coefficients, and are named accordingly.
weights	n vector — *only* for the *wfit* functions.
rank	integer, giving the rank
df.residual	degrees of freedom of residuals
qr	the QR decomposition, see qr.

Fits without any columns or non-zero weights do not have the effects and qr components.

.lm.fit() returns a subset of the above, the qr part unwrapped, plus a logical component pivoted indicating if the underlying QR algorithm did pivot.

Examples

```
set.seed(4242)
n <- 6
p <- 2
slop <- complex(real = 4.23, imaginary = 2.323)
slop2 = complex(real = 2.1, imaginary = -3.9)
interc <- complex(real = 1.4, imaginary = 1.804)
e <- complex(real=rnorm(n)/6, imaginary=rnorm(n)/6)
desmat <- matrix(c(complex(real = rnorm(n * p), imaginary = rnorm(n * p)), rep(1, n)), n, p + 1)
y = desmat %*% c(slop, slop2, interc) + e
lm.fit(desmat, y)
```

---

zmodel.matrix *Generate a Model Matrix (Design Matrix) Using Complex Variables*

Description

A function that somewhat replicates model.matrix(), but accepts complex valued data. It will probably be slower and less efficient, but mostly functional. It cannot handle algebraic expressions in formula.

Usage

```
zmodel.matrix(trms, data, contrasts.arg = NULL)
```

Arguments

trms	A "terms" object as generated by stats::terms().
data	A data frame containing the data referenced by the symbolic formula / model in trms.
contrasts.arg	a list, default is NULL. Not currently supported. See stats::model.matrix() for details.

Value

A model matrix, AKA a design matrix for a regression, containing the relevant information from data.

See Also

stats::model.matrix, lm

Examples

```
set.seed(4242)
slop <- complex(real = 4.23, imaginary = 2.323)
interc <- complex(real = 1.4, imaginary = 1.804)
tframe <- expand.grid(-3:3,-3:3)
Xt <- complex(real = tframe[[1]], imaginary = tframe[[2]])
tframe <- data.frame(Xt=Xt, Yt= Xt * slop + interc + complex(real=rnorm(length(Xt)),
 imaginary=rnorm(length(Xt))))
testterms <- terms(Yt ~ Xt)
zmodel.matrix(testterms, tframe)
```

Index