

Package: complex (via r-universe)

October 7, 2026

Type Package

Title Time Series Analysis and Forecasting Using Complex Variables

Version 1.0.3

Date 2026-09-25

URL <https://github.com/openforecast-org/complex>

BugReports <https://github.com/openforecast-org/complex/issues>

Language en-GB

Description Implements the instruments for complex-valued modelling, including time series analysis and forecasting. This is based on the monograph by Svetunkov & Svetunkov (2024) [<doi:10.1007/978-3-031-62608-1>](https://doi.org/10.1007/978-3-031-62608-1).

License LGPL-2.1

Depends R (>= 3.5.0), greybox (>= 0.5.0), legion

Imports stats, graphics, nloptr, mvtnorm, pracma

LinkingTo Rcpp, RcppArmadillo

Suggests testthat, knitr, rmarkdown

Encoding UTF-8

Roxygen list(old_usage = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake texlive libssl-dev

Repository <https://openforecast-org.r-universe.dev>

Date/Publication 2026-09-26 20:40:21 UTC

RemoteUrl <https://github.com/openforecast-org/complex>

RemoteRef HEAD

RemoteSha 6f1b1de54fa7662a29dc27a32d2c8a94cdd1f2ac

Contents

auto.carima	2
cacf	4
carima	6
clm	8
clog	12
complex2mat	13
cplot	14
cscale	15
cvar	16
dcgnorm	18
dcnorm	19
invert	21
Index	22

auto.carima	<i>Automatic Complex ARIMA</i>
-------------	--------------------------------

Description

Function selects the orders of complex ARIMA (see [carima](#)) based on an information criterion.

Usage

```
auto.carima(y, orders = list(ar = 3, i = 2, ma = 3), constant = NULL,
  ic = c("AICc", "AIC", "BIC", "BICc"), search = c("stepwise", "full"),
  distribution = c("dcnorm", "dcgnorm"), h = 0, holdout = FALSE,
  silent = TRUE, fast = FALSE, ...)
```

Arguments

y	vector or ts object with the complex-valued time series.
orders	list of maximum orders <code>list(ar=p, i=d, ma=q)</code> (or vector <code>c(p,d,q)</code>).
constant	if NULL, the function checks whether the constant is needed. Otherwise TRUE or FALSE.
ic	the information criterion used in the selection.
search	"stepwise" for the neighbourhood search or "full" for all the combinations of orders.
distribution	the distribution used in the likelihood, see clm .
h	the forecast horizon.
holdout	if TRUE, the last h observations are held out.
silent	if FALSE, the progress is printed and the final model is plotted.
fast	if TRUE, the models are screened via the Hannan-Rissanen method before the search (see details). Ignored with <code>search="full"</code> .
...	other parameters passed to carima and clm .

Details

The selection follows the principles of [auto.ssarima](#): the AR and MA orders are selected for each order of differencing, and the best model across all of them is chosen. With `search="stepwise"`:

1. For each d from 0 to `orders$i`, the AR and MA orders are selected via a neighbourhood search (similar to Hyndman & Khandakar, 2008): starting from `cARIMA(0,d,0)`, all models with p and q different by at most one are fitted, and the search moves to the best of them, as long as the information criterion improves. If `constant=NULL`, the constant is included for $d = 0$ and excluded otherwise;
2. If `constant=NULL`, the best model of each d is re-estimated with the opposite option for the constant and the neighbourhood search continues from it. The best of all the estimated models is returned.

With `fast=TRUE`, all `cARIMA(p,d,q)` with and without constant are first estimated via the Hannan-Rissanen method (Hannan & Rissanen, 1982): the differences are regressed on their lags and on the lags of the errors approximated by a long AR, via complex least squares. Their information criteria are approximate. Then the best of them for each d , constant and MA order is estimated via the likelihood, and the neighbourhood search continues from the best of these models (with its d and constant). With heavy tails, the least squares used in the screening is not efficient and tends to underrate the models with MA terms, which is why the best model for each MA order is estimated. This is still an approximation: on wind data with heavy tails, `fast=TRUE` selected a model with an information criterion worse by 16 than the one found with `fast=FALSE`, while being about three times faster.

With `search="full"`, all combinations of p , d , q and the constant are fitted (except for `cARIMA(p,0,q)` without constant when `constant=NULL`).

Models with different orders of differencing and AR use different numbers of initial values, which `clm` constructs from the data. To make the information criteria comparable, all models are estimated on the whole sample, but their criteria are calculated on the common sample without the first k observations, where k is the sum of the maximum orders. The log-likelihood on that sample is calculated at the estimated parameters, with the scale re-estimated.

Every model is estimated from the Hannan-Rissanen estimates of its parameters (see `fast` in `clm`) and from the parameters of the closest model already estimated (with zeros for the new parameters), and the one with the higher likelihood is kept.

Value

The selected model of the class `carima` (see [carima](#)), with additional elements:

- `ICs` - data frame with the orders, the constant and the information criterion (on the common sample) of all the models estimated in the process,
- `ic` - the information criterion used,
- `icObsDropped` - the number of first observations dropped for the criteria,
- `ICsScreening` - with `fast=TRUE`, the approximate criteria of all the models estimated via Hannan-Rissanen.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

References

- Hannan, E. J., Rissanen, J. (1982). Recursive estimation of mixed autoregressive-moving average order. *Biometrika*, 69(1), 81-94.
- Svetunkov, I., Boylan, J. E. (2019). State-space ARIMA for supply-chain forecasting. *International Journal of Production Research*, 58(3), 818-827.
- Hyndman, R. J., Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 27(3), 1-22.

See Also

[carima](#), [auto.ssarima](#), [auto.msarima](#)

Examples

```
set.seed(41)
e <- rcnorm(300, 0, sigma2=1, varsigma2=0.3+0.2i)
y <- cumsum(e + c(0, (0.5-0.2i)*e[-300]))
model <- auto.carima(y, orders=list(ar=2, i=1, ma=2))
model
model$ICs
```

cacf

Complex Correlation Function Estimation

Description

The functions compute (and by default plot) estimates of the Complex Autocovariance, or Complex Autocorrelation, or Partial Complex Autocorrelation functions.

Usage

```
cacf(x, lag.max = NULL, method = c("direct", "conjugate", "pearson",
  "kendall", "spearman"), type = c("correlation", "covariance", "partial"),
  plot = TRUE, ...)
```

```
cpacf(x, lag.max = NULL, method = c("direct", "conjugate", "pearson",
  "kendall", "spearman"), plot = TRUE, ...)
```

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

```
## S3 method for class 'cacf'
plot(x, which = c(1, 2), ask = length(which) > 1,
  level = 0.95, ...)
```

Arguments

<code>x</code>	vector of complex variables.
<code>lag.max</code>	maximum number of lags. See acf for more details.
<code>method</code>	method to use in the calculation of the measure. "conjugate" means that it is based on the multiplication by conjugate number. "direct" means the calculation without the conjugate (i.e. "pseudo" moment). method can also be "pearson", "kendall", or "spearman", defining what correlation coefficient to use after the MDS transformation of complex variables <code>x</code> and <code>y</code> .
<code>type</code>	character string giving the type of cACF to be computed. Allowed values are "correlation" (the default) and "covariance". Will be partially matched.
<code>plot</code>	logical. If TRUE (the default) the cACF is plotted on complex plane and as two linear graphs for real and imaginary parts.
<code>...</code>	Parameter for the <code>plot()</code> function.
<code>which</code>	Determines, which of the plots to produce. 1 is the plot of real and imaginary parts. 2 is the plot of absolute value and the argument.
<code>ask</code>	Determines, whether to ask before producing a new plot or not.
<code>level</code>	Confidence level for the non-rejection region of the correlation coefficient.

Details

For `type="correlation"` and `"covariance"`, the estimates are based on the sample pseudo covariance and use pseudo correlation [ccor](#) and complex covariance [ccov](#) respectively. Note that the function does not calculate values for lag 0. Also, the function will automatically remove NAs. Finally, function does not have `demean` parameter (as, for example, is done in [acf](#)), because `ccov()` and `ccor()` do that automatically.

`cpacf()` produces the partial complex ACF based on complex regression model of variable on its lags.

The generic function `plot` has a method for objects of class "cacf".

The lag is returned and plotted in units of time, and not numbers of observations.

There is a print and plot methods for objects of class "cacf".

Value

An object of class "cacf", which is a list with the following elements:

- `lag` A three dimensional array containing the lags at which the cACF is estimated.
- `acf` An array with the same dimensions as `lag` containing the estimated cACF.
- `method` The method used in calculation (same as the `method` argument).
- `type` The type of correlation (same as the `type` argument).
- `n.used` The number of observations in the time series.
- `series` The name of the series `x`.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

References

- Svetunkov, S. & Svetunkov I. (2022) Complex Autoregressions. In Press.

See Also

[acf](#), [ccor](#)

Examples

```
# Generate random complex variables
x <- complex(real=rnorm(100,10,10), imaginary=rnorm(100,10,10))

# Calculate cACF
cacf(x)
```

carima

Complex ARIMA

Description

Function constructs complex ARIMA(p,d,q) model for a complex-valued time series, without explanatory variables. It is a wrapper of [clm](#).

Usage

```
carima(y, orders = list(ar = 0, i = 0, ma = 0), constant = FALSE,
      arma = NULL, model = NULL, loss = c("likelihood", "OLS", "CLS", "MSE",
      "MAE", "HAM"), distribution = c("dcnorm", "dcgnorm"), h = 0,
      holdout = FALSE, silent = TRUE, ...)

## S3 method for class 'carima'
forecast(object, h = 10, interval = c("none",
  "confidence", "prediction"), level = 0.95, ...)
```

Arguments

y	vector or ts object with the complex-valued time series.
orders	orders of the model: either a vector c(p,d,q) or a list list(ar=p, i=d, ma=q) (elements that are not provided are set to zero).
constant	if TRUE, the constant term is included in the model and estimated. Can also be a complex number (the value of the constant), in which case all the other parameters need to be provided via arma.
arma	list list(ar=..., ma=...) or a vector (AR parameters followed by MA ones) with the fixed complex AR/MA parameters. All of them need to be provided; the model is then not estimated.

model	previously estimated carima model. If provided, its orders, constant, loss, distribution and parameters are used, and the model is applied to y without re-estimation.
loss	the loss function used in the estimation, see clm .
distribution	the distribution used in the likelihood, see clm .
h	the forecast horizon. If positive, the forecasts are produced and returned in forecast.
holdout	if TRUE, the last h observations are held out: the model is estimated on the rest and its forecasts are compared with the held-out values.
silent	if FALSE, the function plots the fit and the forecast at the end.
...	other parameters passed to clm , e.g. shape for distribution="dcgnorm", B or the parameters of the optimiser.
object	the model estimated via carima().
interval	the type of interval, see clm .
level	the confidence level for the prediction interval.

Details

The model is

$$(1 - \phi_1 B - \dots - \phi_p B^p)(1 - B)^d y_t = c + (1 + \theta_1 B + \dots + \theta_q B^q) \epsilon_t,$$

where all the parameters are complex and c is the constant (included only when constant is not FALSE). With $d > 0$ the constant acts as a drift.

Only the non-seasonal model is supported. Parameters are estimated via [clm](#), so its restrictions apply: the first lags are extrapolated and the first MA errors are set to zero (there is no initial and no bounds). Fixed parameters (arma, numeric constant) can only be provided for all of the parameters of the model at once.

Value

An object of the class `c("carima", "clm", "greybox")`: the model returned by [clm](#), plus

- orders - the list of orders,
- constant - whether the constant is included,
- h, holdout - the horizon and the holdout flag,
- forecast - the point forecasts (if $h > 0$),
- holdoutData - the held-out values (if holdout=TRUE),
- accuracy - the error measures on the holdout: real and imaginary parts of the mean error (ME_r, ME_i), MAE (mean modulus of the error) and RMSE (root mean squared modulus of the error).

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

See Also

[clm](#), [msarima](#)

Examples

```
set.seed(41)
e <- rcnorm(200, 0, sigma2=1, varsigma2=0.3+0.2i)
y <- cumsum(e + c(0, 0.5*e[-200]))
model <- carima(y, orders=c(0,1,1), h=10, holdout=TRUE)
summary(model)
model$accuracy
forecast(model, h=5)
```

clm	<i>Complex Linear Model</i>
-----	-----------------------------

Description

Function estimates complex variables model

Usage

```
clm(formula, data, subset, na.action, loss = c("likelihood", "OLS", "CLS",
  "MSE", "MAE", "HAM"), distribution = c("dcnorm", "dcgnorm"),
  orders = c(0, 0, 0), scaling = c("normalisation", "standardisation",
  "max", "none"), parameters = NULL, fast = FALSE, ...)

## S3 method for class 'clm'
sigma(object, type = NULL, ...)

## S3 method for class 'clm'
vcov(object, type = NULL, ...)

## S3 method for class 'clm'
summary(object, level = 0.95, ...)
```

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. Can also include trend, which would add the global trend.
- data a data frame or a matrix, containing the variables in the model.
- 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.
loss	The type of Loss Function used in optimization. loss can be: • OLS - Ordinary Least Squares method, relying on the minimisation of the conjoint variance of the error term; • CLS - Complex Least Squares method, relying on the minimisation of the complex variance of the error term; • likelihood - the model is estimated via the maximisation of the likelihood of the complex Normal distribution; • MSE (Mean Squared Error), • MAE (Mean Absolute Error), • HAM (Half Absolute Moment), A user can also provide their own function here as well, making sure that it accepts parameters actual, fitted and B. Here is an example: <pre>lossFunction <- function(actual, fitted, B, xreg) return(mean(abs(actual-fitted))) loss=lossFunction</pre>
distribution	the distribution used in the likelihood, when loss="likelihood": • "dcnorm" - complex normal distribution (see dcnorm); • "dcgnorm" - complex generalised normal distribution (see dcgnorm). Its shape parameter is estimated, unless it is provided via shape in the ellipsis. The scale and the pseudo-scale are not estimated by the optimiser: the circularity coefficient (pseudo-scale divided by scale) is taken from the moments of the residuals and the scale is the maximum likelihood estimate given that coefficient. This is exact maximum likelihood only for shape=1, and an approximation otherwise.
orders	vector of orders of complex ARIMA(p,d,q).
scaling	NOT YET IMPLEMENTED!!! Defines what type of scaling to do for the variables. See cscale for the explanation of the options.
parameters	vector of parameters of the linear model. When NULL, it is estimated.
fast	if TRUE, then the function won't check whether the data has variability and whether the regressors are correlated. Might cause trouble, especially in cases of multicollinearity. In case of cARIMA without explanatory variables (the intercept is allowed), fast=TRUE also makes the function use the Hannan-Rissanen estimates (Hannan & Rissanen, 1982) of the AR and MA parameters as the starting values of the optimiser (and, for distribution="dcgnorm", the shape that maximises the likelihood of their residuals), instead of the default ones.
...	additional parameters to pass to distribution functions. This includes: • FI=TRUE will make the function also produce Fisher Information matrix, which then can be used to calculate variances of smoothing parameters and initial states of the model. This is used in the vcov method; • shape - the value of the shape parameter of distribution="dcgnorm". If provided, the shape is not estimated;

You can also pass parameters to the optimiser:

1. `B` - the vector of starting values of parameters for the optimiser, should correspond to the explanatory variables. If formula for scale was provided, the parameters for that part should follow the parameters for location; for `distribution="dcgnorm"` with the estimated shape, the starting value of the shape can be added as the last element of `B`;
2. `algorithm` - the algorithm to use in optimisation ("`NLOPT_LN_SBPLX`" by default);
3. `maxeval` - maximum number of evaluations to carry out. Default is 40 per estimated parameter. In case of LASSO / RIDGE the default is 80 per estimated parameter;
4. `maxtime` - stop, when the optimisation time (in seconds) exceeds this;
5. `xtol_rel` - the precision of the optimiser (the default is $1E-6$);
6. `xtol_abs` - the absolute precision of the optimiser (the default is $1E-8$);
7. `ftol_rel` - the stopping criterion in case of the relative change in the loss function (the default is $1E-4$);
8. `ftol_abs` - the stopping criterion in case of the absolute change in the loss function (the default is 0 - not used);
9. `print_level` - the level of output for the optimiser (0 by default). If equal to 41, then the detailed results of the optimisation are returned.

You can read more about these parameters by running the function [nloptr.print.options](#).

<code>object</code>	Object of class "clm" estimated via <code>clm()</code> function.
<code>type</code>	Type of sigma to return. This is calculated based on the residuals of the estimated model and can be "direct", based on the direct variance, "conjugate", based on the conjugate variance and "matrix", returning covariance matrix for the complex error. If NULL then will return value based on the loss used in the estimation: OLS -> "conjugate", CLS -> "direct", likelihood -> "matrix".
<code>level</code>	What confidence level to use for the parameters of the model.

Details

This is a function, similar to [lm](#), but supporting several estimation techniques for complex variables regression.

Value

Function returns `model` - the final model of the class "clm", which contains:

- `coefficients` - estimated parameters of the model,
- `FI` - Fisher Information of parameters of the model. Returned only when `FI=TRUE`,
- `fitted` - fitted values,
- `residuals` - residuals of the model,
- `mu` - the estimated location parameter of the distribution,
- `scale` - the estimated scale parameter of the distribution. If a formula was provided for scale, then an object of class "scale" will be returned.

- logLik - log-likelihood of the model. Only returned, when loss="likelihood" and in a special case of complex least squares.
- loss - the type of the loss function used in the estimation,
- lossFunction - the loss function, if the custom is provided by the user,
- lossValue - the value of the loss function,
- df.residual - number of degrees of freedom of the residuals of the model,
- df - number of degrees of freedom of the model,
- call - how the model was called,
- rank - rank of the model,
- data - data used for the model construction,
- terms - terms of the data. Needed for some additional methods to work,
- B - the value of the optimised parameters. Typically, this is a duplicate of coefficients,
- other - the list of all the other parameters either passed to the function or estimated in the process, but not included in the standard output (e.g. alpha for Asymmetric Laplace),
- timeElapsed - the time elapsed for the estimation of the model.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

References

- Svetunkov, S. & Svetunkov I. (2022) Complex Autoregressions. In Press.

See Also

[alm](#)

Examples

```
### An example with mtcars data and factors
x <- complex(real=rnorm(1000,10,10), imaginary=rnorm(1000,10,10))
a0 <- 10 + 15i
a1 <- 2-1.5i
y <- a0 + a1 * x + 1.5*complex(real=rnorm(length(x),0,1), imaginary=rnorm(length(x),0,1))

complexData <- cbind(y=x,x=x)
complexModel <- clm(y~x, complexData)
summary(complexModel)

plot(complexModel, 7)
```

clog	<i>Functions that transform real and imaginary parts of a complex variable</i>
------	--

Description

Function `clog()` will take logarithm of real and imaginary parts separately and then merge the resulting variable in the complex one. The function `cexp()` does the opposite transform, taking exponent of parts and then merging them.

Usage

```
clog(y, base = exp(1))
```

```
cexp(y, base = exp(1))
```

Arguments

<code>y</code>	vector of a complex variable in the original scale.
<code>base</code>	a positive or complex number: the base with respect to which logarithms/powers are computed. Defaults to <code>exp(1)</code> .

Value

A vector of the same size as `y`, containing transformed complex variable.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

References

- Svetunkov, S. & Svetunkov I. (2022) Complex Autoregressions. In Press.

See Also

[cscale](#)

Examples

```
# Generate random complex variables
y <- complex(real=rnorm(100,100,10), imaginary=rnorm(100,100,10))

yLog <- clog(y)
cexp(yLog)
```

complex2mat

*Functions to manipulate complex variables and matrices***Description**

`complex2mat()` constructs a matrix from the provided complex variable, while `complex2vec()` returns a vector (in mathematical sense), both of them split the real and imaginary parts. `mat2complex()` and `vec2complex()` do the reverse of the respective functions. See details for explanation.

Usage

```
complex2mat(x)
```

```
complex2vec(x)
```

```
mat2complex(x)
```

```
vec2complex(x)
```

Arguments

`x` vector or matrix of complex variables.

Details

Complex variable $x + iy$ can be represented as a vector $(x \ y)'$ or as a matrix: $(x \ -y) \ (y \ x)$

`complex2mat()` returns the latter, while `complex2vec()` returns the former. If a user provides a vector of complex variables, the values are stacked above each other. If a matrix is provided, a higher dimensional matrix is returned.

`mat2complex()` and `vec2complex()` return complex variables based on provided matrix.

The function is needed to calculate some statistics for complex variables in vector form.

Value

A matrix with real and imaginary parts of `x` split into columns (and rows in case of `complex2mat()`).

#' @references

- Svetunkov, S. & Svetunkov I (2022) Complex Autoregressions. In Press.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

References

- Svetunkov, S. & Svetunkov I. (2022) Complex Autoregressions. In Press.

See Also[clm](#)**Examples**

```
# Generate random complex variables
x <- complex(real=rnorm(100,10,10), imaginary=rnorm(100,10,10))

# Get a matrix and a vector for one value
complex2mat(x[1])
complex2vec(x[1])

# Get matrices for all values
complex2mat(x)
complex2vec(x)
```

cplot*Scatterplots for complex variables*

Description

Function produces six scatterplots to show relations between the two complex variables x and y.

Usage

```
cplot(x, y, which = 1, ...)
```

Arguments

x	vector of a complex variable.
y	second vector of a complex variable.
which	defines, what type of plot to produce. which=1 will produce six scatterplots, while which=2 will produce a scatterplot of data after multidimensional scaling (creating projections of complex variables to x and y axes).
...	othehr parameters passed to plot method. Works only for which=2.

Details

The plots are positioned to satisfy two rules: 1. When a scatterplot for a c.r.v. is produced, the real part should be in x-axis, while the imaginary should be in the y-axis. 2. When parts of variables x and y are compared, the part for x should be in x-axis, while the part for y should be in y-axis, which should reflect the idea that x could be an explanatory variable for y.

Value

The function produces a plot and does not return any value

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

References

- Svetunkov, S. & Svetunkov I. (2022) Complex Autoregressions. In Press.

See Also

[ccor](#)

Examples

```
# Generate random complex variables
x <- complex(real=rnorm(100,10,10), imaginary=rnorm(100,10,10))
y <- complex(real=rnorm(100,10,10), imaginary=rnorm(100,10,10))

cplot(x, y)
```

cscale

Functions scale real and imaginary parts of a complex variable

Description

Function `cscale()` will do the scaling based on the selected method, while the function `cdscale()` will transform the variable to get to the original units.

Usage

```
cscale(y, scaling = c("normalisation", "standardisation", "max"))

cdscale(yScaled, y, scaling = c("normalisation", "standardisation", "max"))
```

Arguments

<code>y</code>	vector of a complex variable in the original scale.
<code>scaling</code>	scaling method to use. "normalisation" implies scaling to make sure that <code>y</code> lie in <code>[0, 1]</code> (subtract the minimum value and divide by the range). "standardisation" standardises the variable (i.e. subtract the mean then divide by standard deviation). "max" just divides the variable by the maximum value.
<code>yScaled</code>	vector of the already scaled complex variable.

Value

A vector of the same size as `y`, containing scaled complex variable.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

References

- Svetunkov, S. & Svetunkov I. (2022) Complex Autoregressions. In Press.

See Also

[scale](#)

Examples

```
# Generate random complex variables
y <- complex(real=rnorm(100,10,10), imaginary=rnorm(100,10,10))

yScaled <- cscale(y)
cdescale(yScaled, y)
```

cvar

Correlation, Variance and Covariance (Matrices) for complex variables

Description

Functions `cvar()`, `ccov()` and `ccor()` return respectively complex variance, covariance and correlation based on the provided complex vector/matrix `x`. Function `covar()` returns the covariance matrix based on a complex vector/matrix.

Usage

```
cvar(x, method = c("direct", "conjugate"), df = NULL, ...)

ccov(x, y, method = c("direct", "conjugate"), df = NULL, ...)

ccor(x, y, method = c("direct", "conjugate", "pearson", "kendall",
  "spearman"), ...)

ccov2cor(V)

covar(x, df = NULL)
```

Arguments

<code>x</code>	vector or matrix of complex variables. If it is matrix then the variable <code>y</code> is ignored.
<code>method</code>	method to use in the calculation of the measure. "conjugate" means that it is based on the multiplication by conjugate number. "direct" means the calculation without the conjugate (i.e. "pseudo" moment). For <code>ccor</code> the variable <code>method</code> can also be "pearson", "kendall", or "spearman", defining what correlation coefficient to use after the MDS transformation of complex variables <code>x</code> and <code>y</code> .
<code>df</code>	Number of degrees of freedom to use in the calculation of the statistics.
<code>...</code>	parameters passed to <code>mean()</code> functions. For example, this can be <code>na.rm=TRUE</code> to remove missing values or <code>trim</code> to define the trimming in the mean (see mean).
<code>y</code>	second vector to calculate covariance or correlations with.
<code>V</code>	complex (pseudo)covariance matrix.

Details

Only the parametric correlation is supported by the function. If `x` is matrix, then `y` is ignored.

`covar()` function returns a covariance matrix calculated for the provided complex vector or matrix `x`.

Value

A scalar or a matrix with resulting complex variables.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

References

- Svetunkov, S. & Svetunkov I. (2022) Complex Autoregressions. In Press.

See Also

[cor](#)

Examples

```
# Generate random complex variables
x <- complex(real=rnorm(100,10,10), imaginary=rnorm(100,10,10))
y <- complex(real=rnorm(100,10,10), imaginary=rnorm(100,10,10))

# Create a matrix of complex variables
z <- cbind(x,y)

# Calculate measures
cvar(x)
```

```
cvar(z)
ccor(x,y)
ccor(z)
```

dcgnorm

Complex Generalised Normal Distribution

Description

Density and random generation for the complex generalised normal distribution (also known as the complex generalised Gaussian or the bivariate exponential power distribution) with location μ , scale scale , pseudo-scale pseudoscale and shape shape .

Usage

```
dcgnorm(q, mu = 0, scale = 1, pseudoscale = 0, shape = 1,
        log = FALSE)
```

```
rcgnorm(n = 1, mu = 0, scale = 1, pseudoscale = 0, shape = 1)
```

Arguments

<code>q</code>	vector of quantiles (complex).
<code>mu</code>	vector of location parameters (complex).
<code>scale</code>	scale parameter (real, positive).
<code>pseudoscale</code>	pseudo-scale parameter (complex, with the modulus smaller than scale).
<code>shape</code>	shape parameter (real, positive).
<code>log</code>	if TRUE, then the logarithm of the density is returned.
<code>n</code>	number of observations. Should be a single number.

Details

For the error term $e = z - \mu$, the density is

$$f(z) = \frac{2\beta}{\pi\Gamma(1/\beta)2^{1/\beta}\sqrt{\alpha^4 - |\psi^2|^2}} \exp\left(-\frac{1}{2}Q^\beta\right),$$

where $Q = 2(\alpha^2|e|^2 - \Re(\overline{\psi^2}e^2))/(\alpha^4 - |\psi^2|^2)$, α^2 is the scale, ψ^2 is the pseudo-scale and β is the shape.

With $\text{shape}=1$ this is the complex normal distribution with variance scale and pseudo-variance pseudoscale (see [dcnorm](#)). Values of shape below one give heavier tails ($\text{shape}=0.5$ is Laplace-like), values above one give lighter tails (the uniform distribution on an ellipse in the limit). Note that the shape here applies to the quadratic form, so $\text{shape}=1$ corresponds to $\text{shape}=2$ in [dgnorm](#).

The variance and the pseudo-variance of the distribution equal scale and pseudoscale multiplied by $2^{1/\beta}\Gamma(2/\beta)/(2\Gamma(1/\beta))$.

Random numbers are generated using the fact that Q^β follows the Gamma distribution with shape $1/\beta$ and scale 2, and the angle is uniform in the standardised space.

Value

dcgnorm returns the density, rcgnorm returns a vector of complex random values.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

References

- Novey, M., Adali, T., Roy, A. (2010). A complex generalized Gaussian distribution - characterization, generation, and estimation. IEEE Transactions on Signal Processing, 58(3), 1427-1433.
- Gomez, E., Gomez-Villegas, M. A., Marin, J. M. (1998). A multivariate generalization of the power exponential family of distributions. Communications in Statistics - Theory and Methods, 27(3), 589-600.

See Also

[dcnorm](#)

Examples

```
x <- rcgnorm(1000, 1+1i, scale=2, pseudoscale=0.6+0.8i, shape=0.5)
plot(x)
dcgnorm(x[1:5], 1+1i, scale=2, pseudoscale=0.6+0.8i, shape=0.5)
```

dcnorm

Complex Normal Distribution

Description

Density, cumulative distribution, quantile functions and random number generation for the Complex Normal distribution.

Usage

```
dcnorm(q, mu = 0, sigma2 = 1, varsigma2 = 0, log = FALSE, ...)

pcnorm(lower = -Inf, upper = Inf, mu = 0, sigma2 = 1, varsigma2 = 0,
...)

qcnorm(p, mu = 0, sigma2 = 1, varsigma2 = 0, ...)

rcnorm(n = 1, mu = 0, sigma2 = 1, varsigma2 = 0, ...)
```

Arguments

q	vector of quantiles.
mu	vector of location parameters (means).
sigma2	vector of conjugate variances.
varsigma2	vector of direct variances.
log	if TRUE, then probabilities are returned in logarithms.
...	Other parameters passed to the mvtnorm functions.
lower	complex number of lower limits of length n.
upper	complex number of upper limits of length n.
p	vector of probabilities.
n	number of observations. Should be a single number.

Details

Complex Normal distribution is a special case of a multivariate normal distribution, which is parametrised using direct and conjugate variances instead of the covariance matrix.

These functions are just wrappers for the functions from the mvtnorm package.

Note that sigma2 and varsigma2 are the conjugate and direct variances, not the standard deviations!

Both pcnorm and qcnorm are returned for the lower tail of the distribution.

All the functions are defined for non-negative values only.

Value

Depending on the function, various things are returned (usually either vector or scalar):

- dcnorm returns the density function values for the provided parameters, based on [Mvnorm](#) function.
- pcnorm returns the values of the cumulative function for the provided parameters, based on [pmvnorm](#) function.
- qcnorm returns quantiles of the distribution, based on [qmvnorm](#) function.
- rcnorm returns a vector of random variables generated from the Complex Normal distribution, based on [Mvnorm](#) function.

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

Examples

```
dcnorm(89+90i, 100+100i, 2, 1+1i)
pcnorm(90+90i, 110+110i, 100+100i, 2, 1+1i)
qcnorm(0.95, 100+100i, 2, 1+1i)
rcnorm(1000, 100+100i, 2, 1+1i)
```

`invert`*Function calculates inverse of matrix of complex variables*

Description

The function accepts a square complex matrix and returns inverse of it.

Usage

```
invert(x)
```

Arguments

`x` The square matrix of complex variables.

Value

The function returns a matrix of the same size as the original matrix `x`

Author(s)

Ivan Svetunkov, <ivan@svetunkov.com>

See Also

[solve](#)

Examples

```
invert(matrix(complex(real=c(1,2), imaginary=c(1.1,2.1)), 2, 2))
```

Index

- * **distribution**
 - dcgnorm, 18
 - dcnorm, 19
- * **models**
 - auto.carima, 2
 - carima, 6
 - clm, 8
- * **nonlinear**
 - auto.carima, 2
 - carima, 6
 - clm, 8
- * **regression**
 - auto.carima, 2
 - carima, 6
 - clm, 8
- * **ts**
 - auto.carima, 2
 - carima, 6
 - clm, 8
- * **univar**
 - cacf, 4
 - clog, 12
 - complex2mat, 13
 - cplot, 14
 - cscale, 15
 - cvar, 16
- acf, 5, 6
- alm, 11
- auto.carima, 2
- auto.msarima, 4
- auto.ssarima, 3, 4
- cacf, 4
- carima, 2–4, 6
- ccor, 5, 6, 15
- ccor (cvar), 16
- ccov, 5
- ccov (cvar), 16
- ccov2cor (cvar), 16
- cdescale (cscale), 15
- cexp (clog), 12
- cgnormal (dcgnorm), 18
- clm, 2, 3, 6, 7, 8, 8, 14
- clog, 12
- cnormal (dcnorm), 19
- complex2mat, 13
- complex2vec (complex2mat), 13
- cor, 17
- covar (cvar), 16
- cpacf (cacf), 4
- cplot, 14
- cscale, 9, 12, 15
- cvar, 16
- dcgnorm, 9, 18
- dcnorm, 9, 18, 19, 19
- dgnorm, 18
- forecast.carima (carima), 6
- invert, 21
- lm, 10
- mat2complex (complex2mat), 13
- mean, 17
- msarima, 8
- Mvnorm, 20
- na.exclude, 9
- na.fail, 9
- na.omit, 9
- nloptr.print.options, 10
- options, 9
- pcnorm (dcnorm), 19
- plot.cacf (cacf), 4
- pmvnorm, 20
- print.cacf (cacf), 4

qcnorm(dcnorm), [19](#)
qmvnorm, [20](#)

rcgnorm(dcgncnorm), [18](#)
rcnorm(dcnorm), [19](#)

scale, [16](#)
sigma.clm(c1m), [8](#)
solve, [21](#)
summary.clm(c1m), [8](#)

vcov, [9](#)
vcov.clm(c1m), [8](#)
vec2complex(complex2mat), [13](#)