Package: complex (via r-universe)

October 24, 2026

Type Package

Title Time Series Analysis and Forecasting Using Complex Variables

Version 1.0.2

Date 2026-02-03

URL https://github.com/config-i1/complex

BugReports https://github.com/config-i1/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>.

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

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation yes

Author Ivan Svetunkov [aut, cre] (Lecturer at Centre for Marketing
Analytics and Forecasting, Lancaster University, UK)

Maintainer Ivan Svetunkov <ivan@svetunkov.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-02-04 19:30:08 UTC

RemoteUrl https://github.com/cran/complex

RemoteRef HEAD

RemoteSha 97ad74fbfc674ec622c6091bf6708e90c6aaa286

Contents

cacf 2
clm 4
clog 7
complex2mat 8
cplot 9
cscale 10
cvar 11
dcnorm 13
invert 14

Index **16**

`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

`x`	vector of complex variables.
`lag.max`	maximum number of lags. See acf for more details.
`method`	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 `x` and `y`.

type	character string giving the type of cACF to be computed. Allowed values are "correlation" (the default) and "covariance". Will be partially matched.
plot	logical. If TRUE (the default) the cACF is plotted on complex plane and as two linear graphs for real and imaginary parts.
...	Parameter for the plot() function.
which	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.
ask	Determines, whether to ask before producing a new plot or not.
level	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)
```

---

clm *Complex Linear Model*

Description

Function estimates complex variables model

Usage

```
clm(formula, data, subset, na.action, loss = c("likelihood", "OLS", "CLS",
  "MSE", "MAE", "HAM"), 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:

```
lossFunction <- function(actual, fitted, B, xreg) return(mean(abs(actual-fitted)))
loss=lossFunction
```

`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.

`...` Other parameters passed to internal functions.

`object` Object of class "clm" estimated via `clm()` function.

`type` 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".

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

plot(complexModel, 7)
```

clog *Functions that transform real and imaginary parts of a complex variable*

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

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

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).
...	otehr 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 the 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 cdescale() will transform the variable to get to the original units.

Usage

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

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

Arguments

y	vector of a complex variable in the original scale.
scaling	scaling method to use. "normalisation" implies scaling to make sure that y lie in [0, 1] (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.
yScaled	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

x	vector or matrix of complex variables. If it is matrix then the variable y is ignored.
method	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 ccor the variable method can also be "pearson", "kendall", or "spearman", defining what correlation coefficient to use after the MDS transformation of complex variables x and y.
df	Number of degrees of freedom to use in the calculation of the statistics.
...	parameters passed to mean() functions. For example, this can be na.rm=TRUE to remove missing values or trim to define the trimming in the mean (see mean).
y	second vector to calculate covariance or correlations with.
V	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)
```

---

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**
  dcnorm, 13
∗ **models**
  clm, 4
∗ **nonlinear**
  clm, 4
∗ **regression**
  clm, 4
∗ **ts**
  clm, 4
∗ **univar**
  cacf, 2
  clog, 7
  complex2mat, 8
  cplot, 9
  cscale, 10
  cvar, 11

acf, *2*, *3*
alm, *6*

cacf, 2
ccor, *3*, *10*
ccor *(*cvar*)*, 11
ccov, *3*
ccov *(*cvar*)*, 11
ccov2cor *(*cvar*)*, 11
cdescale *(*cscale*)*, 10
cexp *(*clog*)*, 7
clm, 4, *9*
clog, 7
cnormal *(*dcnorm*)*, 13
complex2mat, 8
complex2vec *(*complex2mat*)*, 8
cor, *12*
covar *(*cvar*)*, 11
cpacf *(*cacf*)*, 2
cplot, 9
cscale, *5*, *7*, 10
cvar, 11

dcnorm, 13

invert, 14

lm, *5*

mat2complex *(*complex2mat*)*, 8
mean, *12*
Mvnorm, *14*

na.exclude, *4*
na.fail, *4*
na.omit, *4*

options, *4*

pcnorm *(*dcnorm*)*, 13
plot.cacf *(*cacf*)*, 2
pmvnorm, *14*
print.cacf *(*cacf*)*, 2

qcnorm *(*dcnorm*)*, 13
qmvnorm, *14*

rcnorm *(*dcnorm*)*, 13

scale, *11*
sigma.clm *(*clm*)*, 4
solve, *15*
summary.clm *(*clm*)*, 4

vcov.clm *(*clm*)*, 4
vec2complex *(*complex2mat*)*, 8