# Package: growth (via r-universe)

September 5, 2024

**Version** 1.1.1

**Title** Multivariate Normal and Elliptically-Contoured Repeated Measurements Models

**Depends** R (>= 1.4), rmutil

**Description** Functions for fitting various normal theory (growth curve) and elliptically-contoured repeated measurements models with ARMA and random effects dependence.

**License** GPL (>= 2)

**URL** http://www.commanster.eu/rcode.html

**BugReports** https://github.com/swihart/growth/issues

**Encoding** UTF-8

**LazyData** true

**LazyLoad** true

**RoxygenNote** 6.1.1

**Repository** https://swihart.r-universe.dev

**RemoteUrl** https://github.com/swihart/growth

**RemoteRef** HEAD

**RemoteSha** c46451dd8d4cf0e933feb82df7a5ecd1c2bee0f1

## Contents

carma *Continuous ARMA for Unequally Spaced Repeated Measurements*

## Description

carma is designed to handle a polynomial within subject design matrix with unequally spaced observations which can be at different times for different subjects. The origin of time is taken as the mean time of all the subjects. The within subject errors are assumed to be independent Gaussian or have a continuous time ARMA(p,q) Gaussian structure with the option to include measurement error. The between subject random coefficients are assumed to have an arbitrary covariance matrix. The fixed effect design matrix is a polynomial of equal or higher order than the within subject design matrix. This matrix can be augmented by covariates multiplied by polynomial design matrices of any order up to the order of the first partition of the design matrix. The method is based on exact maximum likelihood using the Kalman filter to calculate the likelihood.

## Usage

```
carma(response = NULL, ccov = NULL, times = NULL, torder = 0,
  interaction, arma = c(0, 0, 0), parma = NULL, pre = NULL,
  position = NULL, iopt = TRUE, resid = TRUE,
  transform = "identity", delta = NULL, envir = parent.frame(),
  print.level = 0, typsize = abs(p), ndigit = 10, gradtol = 1e-05,
  steptol = 1e-05, iterlim = 100, fscale = 1, stepmax = 10 * sqrt(p
  %*% p))

## S3 method for class 'carma'
coef(object, ...)

## S3 method for class 'carma'
deviance(object, ...)

## S3 method for class 'carma'
residuals(object, recursive = TRUE, ...)

## S3 method for class 'carma'
print(x, digits = max(3, .Options$digits - 3),
  correlation = TRUE, ...)

## S3 method for class 'carma'
mprofile(z, times = NULL, ccov, plotse = TRUE, ...)
```

## Arguments

response A list of two column matrices with response values and times for each individual, one matrix or dataframe of response values, or an object of either class, response (created by restovec) or repeated (created by rmna or lvna). If the

| | |
|---|---|
| | `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here. |
| `ccov` | A matrix of columns of baseline covariates with one row per individual, a model formula using vectors of the same size, or an object of class, `tccov` (created by `tcctomat`). If response has class, `repeated`, the covariates must be specified as a Wilkinson and Rogers formula unless none are to be used. |
| `times` | When response is a matrix, a vector of possibly unequally spaced times when they are the same for all individuals or a matrix of times. Not necessary if equally spaced. Ignored if response has class, `response` or `repeated`. |
| `torder` | Order of the polynomial in time to be fitted. |
| `interaction` | Vector indicating order of interactions of covariates with time. |
| `arma` | Vector of three values: order of AR, order of MA, binary indicator for presence of measurement error. Not required for an AR(1) if an initial estimate is supplied. If only one value is supplied, it is assumed to be the order of the AR. |
| `parma` | Initial estimates of ARMA parameters. For example, with `arma=c(1,0,0)`, an AR(1), the parameter is `parma[1]=log(theta)`, where `theta` is the positive, continuous time autoregressive coefficient. The finite step autoregression coefficient for a step of length `delta` is then `alpha=exp(-delta*theta)` i.e. `alpha=exp(-delta*exp(parma[1]))`. |
| `pre` | Initial estimates of random effect parameters. |
| `position` | Two column matrix with rows giving index positions of random effects in the covariance matrix. |
| `iopt` | TRUE if optimization should be performed. |
| `resid` | TRUE if residuals to be calculated. |
| `transform` | Transformation of the response variable: `identity`, `exp`, `square`, `sqrt`, or `log`. |
| `delta` | Scalar or vector giving the unit of measurement for each response value, set to unity by default. For example, if a response is measured to two decimals, `delta=0.01`. Ignored if response has class, `response` or `repeated`. |
| `envir` | Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `response`. If `response` has class `repeated`, it is used as the environment. |
| `print.level` | Arguments for nlm. |
| `typsize` | Arguments for nlm. |
| `ndigit` | Arguments for nlm. |
| `gradtol` | Arguments for nlm. |
| `steptol` | Arguments for nlm. |
| `iterlim` | Arguments for nlm. |
| `fscale` | Arguments for nlm. |
| `stepmax` | Arguments for nlm. |
| `object` | An object of class, `carma`. |

| | |
|---|---|
| `...` | additional arguments. |
| `recursive` | If TRUE, recursive residuals or fitted values are given; otherwise, marginal ones. |
| `x` | An object of class, `carma`. |
| `digits` | number of digits to print. |
| `correlation` | logical; print correlations. |
| `z` | An object of class, `carma`. |
| `plotse` | Plot the standard errors around the marginal profile curve. |

## Details

For clustered (non-longitudinal) data, where only random effects will be fitted, `times` are not necessary.

Marginal and individual profiles can be plotted using `mprofile` and `iprofile` and residuals with `plot.residuals`.

For any ARMA of order superior to an AR(1), the (complex) roots of the characteristic equation are printed out; see Jones and Ackerson (1991) for their use in calculation of the covariance function.

## Value

A list of class `carma` is returned that contains all of the relevant information calculated, including error codes.

## Methods (by generic)

- `coef`: Coefficients
- `deviance`: Deviance
- `residuals`: Residuals
- `print`: Print method
- `mprofile`: Special marginal profiles with SEs

## Author(s)

R.H. Jones and J.K. Lindsey

## References

Jones, R. H. and Ackerson, L. M. (1991) Serial correlation in unequally spaced longitudinal data. Biometrika, 77, 721-731.

Jones, R.H. (1993) Longitudinal Data Analysis with Serial Correlation: A State-space Approach. Chapman and Hall

## See Also

`elliptic`, `gar`, `gnlmix`, `glmm`, `gnlmm`, `iprofile`, `kalseries`, `mprofile`, `plot.residuals`, `potthoff`, `read.list`, `restovec`, `rmna`, `tcctomat`, `tvctomat`.

## Examples

```
y <- matrix(rnorm(40),ncol=5)
x1 <- gl(2,4)
x2 <- gl(2,1,8)
# independence with time trend
carma(y, ccov=~x1, torder=2)
# AR(1)
carma(y, ccov=~x1, torder=2, arma=c(1,0,0), parma=-0.5)
carma(y, ccov=~x1, torder=3, interact=3, arma=c(1,0,0), parma=-1)
# ARMA(2,1)
carma(y, ccov=~x1+x2, interact=c(2,0), torder=3,arma=c(2,1,0),
 parma=c(0.3,2,0.7))
# random intercept
carma(y, ccov=~x1+x2, interact=c(2,0), torder=3, pre=-0.4,
 position=c(1,1))
# random coefficients
carma(y, ccov=~x1+x2, interact=c(2,0), torder=3, pre=c(-0.4,0.1),
 position=rbind(c(1,1),c(2,2)))
```

---

`corgram` *Calculate and Plot a Correlogram*

---

## Description

`corgram` calculates the values of a correlogram (autocorrelation function or ACF) and plots it.

## Usage

```
corgram(y, wt = 1, maxlag = NULL, partial = FALSE, add = FALSE,
  lty = 1, xlim = NULL, ylim = NULL, xlab = NULL, ylab = NULL,
  main = NULL, ...)
```

## Arguments

| | |
|---|---|
| `y` | A time series vector. |
| `wt` | Indicator vector with zeros for values to be ignored. |
| `maxlag` | Maximum number of lags for which the correlation is to be calculated. |
| `partial` | If TRUE, the partial autocorrelation function (PACF) is plotted. |
| `add` | If TRUE, adds a new correlogram to an existing plot. |
| `lty` | Plotting parameters |
| `xlim` | Plotting parameters |
| `ylim` | Plotting parameters |
| `xlab` | Plotting parameters |
| `ylab` | Plotting parameters |
| `main` | Plotting parameters |
| `...` | Plotting parameters |

## Value

`corgram` returns a two-column matrix containing the (partial) correlogram coordinates.

## Author(s)

J.K. Lindsey

## Examples

```
y <- rnorm(100)
corgram(y)
corgram(y, partial=TRUE)
```

---

# elliptic — *Nonlinear Multivariate Elliptically-contoured Repeated Measurements Models with AR(1) and Two Levels of Variance Components*

---

## Description

`elliptic` fits special cases of the multivariate elliptically-contoured distribution, the multivariate normal, Student t, and power exponential distributions. The latter includes the multivariate normal (power=1), a multivariate Laplace (power=0.5), and the multivariate uniform (power -> infinity) distributions as special cases. As well, another form of multivariate skew Laplace distribution is also available.

## Usage

```
elliptic(response = NULL, model = "linear", distribution = "normal",
  times = NULL, dose = NULL, ccov = NULL, tvcov = NULL,
  nest = NULL, torder = 0, interaction = NULL,
  transform = "identity", link = "identity",
  autocorr = "exponential", pell = NULL, preg = NULL, covfn = NULL,
  pvar = var(y), varfn = NULL, par = NULL, pre = NULL,
  delta = NULL, shfn = FALSE, common = FALSE, twins = FALSE,
  envir = parent.frame(), print.level = 0, ndigit = 10,
  gradtol = 1e-05, steptol = 1e-05, iterlim = 100, fscale = 1,
  stepmax = 10 * sqrt(theta %*% theta), typsize = abs(c(theta)))

## S3 method for class 'elliptic'
deviance(object, ...)

## S3 method for class 'elliptic'
fitted(object, recursive = FALSE, ...)

## S3 method for class 'elliptic'
residuals(object, recursive = FALSE, ...)
```

```
## S3 method for class 'elliptic'
print(x, digits = max(3, .Options$digits - 3),
  correlation = TRUE, ...)
```

## Arguments

`response` A list of two or three column matrices with response values, times, and possibly nesting categories, for each individual, one matrix or dataframe of response values, or an object of class, `response` (created by `restovec`) or `repeated` (created by `rmna` or `lvna`). If the `repeated` data object contains more than one response variable, give that object in `envir` and give the name of the response variable to be used here.

`model` The model to be fitted for the location. Builtin choices are (1) `linear` for linear models with time-varying covariate; if `torder > 0`, a polynomial in time is automatically fitted; (2) `logistic` for a four-parameter logistic growth curve; (3) `pkpd` for a first-order one-compartment pharmacokinetic model. Otherwise, set this to a function of the parameters or a formula beginning with ~, specifying either a linear regression function for the location parameter in the Wilkinson and Rogers notation or a general function with named unknown parameters that describes the location, returning a vector the same length as the number of observations, in which case `ccov` and `tvcov` cannot be used.

`distribution` Multivariate `normal`, `power exponential`, `Student t`, or skew `Laplace` distribution. The latter is not an elliptical distribution. Note that the latter has a different parametrization of the skew (family) parameter than the univariate skew Laplace distribution in `dskewlaplace`: $skew = \frac{\sigma(1-\nu^2)}{\sqrt{2}\nu}$. Here, zero skew yields a symmetric distribution.

`times` When `response` is a matrix, a vector of possibly unequally spaced times when they are the same for all individuals or a matrix of times. Not necessary if equally spaced. Ignored if response has class, `response` or `repeated`.

`dose` A vector of dose levels for the `pkpd model`, one per individual.

`ccov` A vector or matrix containing time-constant baseline covariates with one line per individual, a model formula using vectors of the same size, or an object of class, `tccov` (created by `tcctomat`). If response has class, `repeated`, with a `linear`, `logistic`, or `pkpd` model, the covariates must be specified as a Wilkinson and Rogers formula unless none are to be used. For the `pkpd` and `logistic` models, all variables must be binary (or factor variables) as different values of all parameters are calculated for all combinations of these variables (except for the logistic model when a time-varying covariate is present). It cannot be used when model is a function.

`tvcov` A list of vectors or matrices with time-varying covariates for each individual (one column per variable), a matrix or dataframe of such covariate values (if only one covariate), or an object of class, `tvcov` (created by `tvctomat`). If times are not the same as for responses, the list can be created with `gettvc`. If response has class, `repeated`, with a `linear`, `logistic`, or `pkpd` model, the covariates must be specified as a Wilkinson and Rogers formula unless none are to be used. Only one time-varying covariate is allowed except for the `linear` model; if more are required, set model equal to the appropriate mean function. This argument cannot be used when model is a function.

nest
: When response is a matrix, a vector of length equal to the number of responses per individual indicating which responses belong to which nesting category. Categoriess must be consecutive increasing integers. This option should always be specified if nesting is present. Ignored if response has class, repeated.

torder
: When the linear model is chosen, order of the polynomial in time to be fitted.

interaction
: Vector of length equal to the number of time-constant covariates, giving the levels of interactions between them and the polynomial in time in the linear model.

transform
: Transformation of the response variable: identity, exp, square, sqrt, or log.

link
: Link function for the location: identity, exp, square, sqrt, or log. For the linear model, if not the identity, initial estimates of the regression parameters must be supplied (intercept, polynomial in time, time-constant covariates, time-varying covariates, in that order).

autocorr
: The form of the autocorrelation function: exponential is the usual $\rho^{|t_i-t_j|}$; gaussian is $\rho^{(t_i-t_j)^2}$; cauchy is $1/(1+\rho(t_i-t_j)^2)$; spherical is $((|t_i-t_j|\rho)^3-3|t_i-t_j|\rho+2)/2$ for $|t_i-t_j|\leq 1/\rho$ and zero otherwise; IOU is the integrated Ornstein-Uhlenbeck process, $(2\rho\min(t_i,t_j)+\exp(-\rho t_i)+\exp(-\rho t_j)-1-\exp(\rho|ti-t_j|))/2\rho^3$.

pell
: Initial estimate of the power parameter of the multivariate power exponential distribution, of the degrees of freedom parameter of the multivariate Student t distribution, or of the asymmetry parameter of the multivariate Laplace distribution. If not supplied for the latter, asymmetry depends on the regression equation in model.

preg
: Initial parameter estimates for the regression model. Only required for linear model if the link is not the identity or a variance (dispersion) function is fitted.

covfn
: Either a function or a formula beginning with ~, specifying how the covariance depends on covariates: either a linear regression function in the Wilkinson and Rogers notation or a general function with named unknown parameters.

pvar
: Initial parameter estimate for the variance or dispersion. If more than one value is provided, the log variance/dispersion depends on a polynomial in time. With the pkpd model, if four values are supplied, a nonlinear regression for the variance/dispersion is fitted.

varfn
: The builtin variance (dispersion) function has the variance/dispersion proportional to a function of the location: pvar*v(mu) = identity or square. If pvar contains two initial values, an additive constant is included: pvar(1)+pvar(2)*v(mu). Otherwise, either a function or a formula beginning with ~, specifying either a linear regression function in the Wilkinson and Rogers notation or a general function with named unknown parameters for the log variance can be supplied. If it contains unknown parameters, the keyword mu may be used to specify a function of the location parameter.

par
: If supplied, an initial estimate for the autocorrelation parameter.

| | |
|---|---|
| `pre` | Zero, one or two parameter estimates for the variance components, depending on the number of levels of nesting. If covfn is specified, this contains the initial estimates of the regression parameters. |
| `delta` | Scalar or vector giving the unit of measurement for each response value, set to unity by default. For example, if a response is measured to two decimals, `delta=0.01`. Ignored if response has class, `response` or `repeated`. |
| `shfn` | If TRUE, the supplied variance (dispersion) function depends on the mean function. The name of this mean function must be the last argument of the variance/dispersion function. |
| `common` | If TRUE, `mu` and `varfn` must both be either functions with, as argument, a vector of parameters having some or all elements in common between them so that indexing is in common between them or formulae with unknowns. All parameter estimates must be supplied in `preg`. If FALSE, parameters are distinct between the two functions and indexing starts at one in each function. |
| `twins` | Only possible when there are two observations per individual (e.g. twin data). If TRUE and `covfn` is supplied, allows the covariance to vary across pairs of twins with the diagonal "variance" of the covariance matrix remaining constant. |
| `envir` | Environment in which model formulae are to be interpreted or a data object of class, `repeated`, `tccov`, or `tvcov`; the name of the response variable should be given in `response`. If `response` has class `repeated`, it is used as the environment. |
| `print.level` | Arguments for nlm. |
| `ndigit` | Arguments for nlm. |
| `gradtol` | Arguments for nlm. |
| `steptol` | Arguments for nlm. |
| `iterlim` | Arguments for nlm. |
| `fscale` | Arguments for nlm. |
| `stepmax` | Arguments for nlm. |
| `typsize` | Arguments for nlm. |
| `object` | An object of class, `elliptic`. |
| `...` | additional arguments. |
| `recursive` | If TRUE, recursive residuals or fitted values are given; otherwise, marginal ones. In all cases, raw residuals are returned, not standardized by the standard deviation (which may be changing with covariates or time). |
| `x` | An object of class, `elliptic`. |
| `digits` | number of digits to print. |
| `correlation` | logical; print correlations. |

## Details

With two levels of nesting, the first is the individual and the second will consist of clusters within individuals.

For clustered (non-longitudinal) data, where only random effects will be fitted, `times` are not necessary.

This function is designed to fit linear and nonlinear models with time-varying covariates observed at arbitrary time points. A continuous-time AR(1) and zero, one, or two levels of nesting can be handled. Recall that zero correlation (all zeros off-diagonal in the covariance matrix) only implies independence for the multivariate normal distribution.

Nonlinear regression models can be supplied as formulae where parameters are unknowns in which case factor variables cannot be used and parameters must be scalars. (See `finterp`.)

Recursive fitted values and residuals are only available for the multivariate normal distribution with a linear model without a variance function and with either an AR(1) of `exponential` form and/or one level of random effect. In these cases, marginal and individual profiles can be plotted using `mprofile` and `iprofile` and residuals with `plot.residuals`.

### Value

A list of class `elliptic` is returned that contains all of the relevant information calculated, including error codes.

### Methods (by generic)

- `deviance`: Deviance method
- `fitted`: Fitted method
- `residuals`: Residuals method
- `print`: Print method

### Author(s)

J.K. Lindsey

### References

Lindsey, J.K. (1999) Multivariate elliptically-contoured distributions for repeated measurements. Biometrics 55, 1277-1280.

Kotz, S., Kozubowski, T.J., and Podgorski, K. (2001) The Laplace Distribution and Generalizations. A Revisit with Applications to Communications, Economics, Engineering, and Finance. Basel: Birkhauser, Ch. 6.

### See Also

`carma`, `dpowexp`, `dskewlaplace`, `finterp`, `gar`, `gettvc`, `gnlmix`, `glmm`, `gnlmm`, `gnlr`, `iprofile`, `kalseries`, `mprofile`, `potthoff`, `read.list`, `restovec`, `rmna`, `tcctomat`, `tvctomat`.

### Examples

```
# linear models
y <- matrix(rnorm(40),ncol=5)
x1 <- gl(2,4)
x2 <- gl(2,1,8)
```

```
# independence with time trend
elliptic(y, ccov=~x1, torder=2)
# AR(1)
elliptic(y, ccov=~x1, torder=2, par=0.1)
elliptic(y, ccov=~x1, torder=3, interact=3, par=0.1)
# random intercept
elliptic(y, ccov=~x1+x2, interact=c(2,0), torder=3, pre=2)
#
# nonlinear models
time <- rep(1:20,2)
dose <- c(rep(2,20),rep(5,20))
mu <- function(p) exp(p[1]-p[3])*(dose/(exp(p[1])-exp(p[2]))*
 (exp(-exp(p[2])*time)-exp(-exp(p[1])*time)))
shape <- function(p) exp(p[1]-p[2])*time*dose*exp(-exp(p[1])*time)
conc <- matrix(rnorm(40,mu(log(c(1,0.3,0.2))),sqrt(shape(log(c(0.1,0.4))))),
 ncol=20,byrow=TRUE)
conc[,2:20] <- conc[,2:20]+0.5*(conc[,1:19]-matrix(mu(log(c(1,0.3,0.2))),
 ncol=20,byrow=TRUE)[,1:19])
conc <- ifelse(conc>0,conc,0.01)
# with builtin function
# independence
elliptic(conc, model="pkpd", preg=log(c(0.5,0.4,0.1)), dose=c(2,5))
# AR(1)
elliptic(conc, model="pkpd", preg=log(c(0.5,0.4,0.1)), dose=c(2,5),
 par=0.1)
# add variance function
elliptic(conc, model="pkpd", preg=log(c(0.5,0.4,0.1)), dose=c(2,5),
 par=0.1, varfn=shape, pvar=log(c(0.5,0.2)))
# multivariate power exponential distribution
elliptic(conc, model="pkpd", preg=log(c(0.5,0.4,0.1)), dose=c(2,5),
 par=0.1, varfn=shape, pvar=log(c(0.5,0.2)), pell=1,
 distribution="power exponential")
# multivariate Student t distribution
elliptic(conc, model="pkpd", preg=log(c(0.5,0.4,0.1)), dose=c(2,5),
 par=0.1, varfn=shape, pvar=log(c(0.5,0.2)), pell=5,
 distribution="Student t")
# multivariate Laplace distribution
elliptic(conc, model="pkpd", preg=log(c(0.5,0.4,0.1)), dose=c(2,5),
 par=0.1, varfn=shape, pvar=log(c(0.5,0.2)),
 distribution="Laplace")
# or equivalently with user-specified function
# independence
elliptic(conc, model=mu, preg=log(c(0.5,0.4,0.1)))
# AR(1)
elliptic(conc, model=mu, preg=log(c(0.5,0.4,0.1)), par=0.1)
# add variance function
elliptic(conc, model=mu, preg=log(c(0.5,0.4,0.1)), par=0.1,
 varfn=shape, pvar=log(c(0.5,0.2)))
# multivariate power exponential distribution
elliptic(conc, model=mu, preg=log(c(0.5,0.4,0.1)), par=0.1,
 varfn=shape, pvar=log(c(0.5,0.2)), pell=1,
 distribution="power exponential")
# multivariate Student t distribution
```

```
elliptic(conc, model=mu, preg=log(c(0.5,0.4,0.1)), par=0.1,
 varfn=shape, pvar=log(c(0.5,0.2)), pell=5,
 distribution="Student t")
# multivariate Laplace distribution
elliptic(conc, model=mu, preg=log(c(0.5,0.4,0.1)), par=0.1,
 varfn=shape, pvar=log(c(0.5,0.2)), pell=5,
 distribution="Laplace")
# or with user-specified formula
# independence
elliptic(conc, model=~exp(absorption-volume)*
 dose/(exp(absorption)-exp(elimination))*
 (exp(-exp(elimination)*time)-exp(-exp(absorption)*time)),
 preg=list(absorption=log(0.5),elimination=log(0.4),
 volume=log(0.1)))
# AR(1)
elliptic(conc, model=~exp(absorption-volume)*
 dose/(exp(absorption)-exp(elimination))*
 (exp(-exp(elimination)*time)-exp(-exp(absorption)*time)),
 preg=list(absorption=log(0.5),elimination=log(0.4),volume=log(0.1)),
 par=0.1)
# add variance function
elliptic(conc, model=~exp(absorption-volume)*
 dose/(exp(absorption)-exp(elimination))*
 (exp(-exp(elimination)*time)-exp(-exp(absorption)*time)),
 preg=list(absorption=log(0.5),elimination=log(0.4),volume=log(0.1)),
 varfn=~exp(b1-b2)*time*dose*exp(-exp(b1)*time),
 par=0.1, pvar=list(b1=log(0.5),b2=log(0.2)))
# variance as function of the mean
elliptic(conc, model=~exp(absorption-volume)*
 dose/(exp(absorption)-exp(elimination))*
 (exp(-exp(elimination)*time)-exp(-exp(absorption)*time)),
 preg=list(absorption=log(0.5),elimination=log(0.4),volume=log(0.1)),
 varfn=~d*log(mu),shfn=TRUE,par=0.1, pvar=list(d=1))
# multivariate power exponential distribution
elliptic(conc, model=~exp(absorption-volume)*
 dose/(exp(absorption)-exp(elimination))*
 (exp(-exp(elimination)*time)-exp(-exp(absorption)*time)),
 preg=list(absorption=log(0.5),elimination=log(0.4),volume=log(0.1)),
 varfn=~exp(b1-b2)*time*dose*exp(-exp(b1)*time),
 par=0.1, pvar=list(b1=log(0.5),b2=log(0.2)), pell=1,
 distribution="power exponential")
# multivariate Student t distribution
elliptic(conc, model=~exp(absorption-volume)*
 dose/(exp(absorption)-exp(elimination))*
 (exp(-exp(elimination)*time)-exp(-exp(absorption)*time)),
 preg=list(absorption=log(0.5),elimination=log(0.4),volume=log(0.1)),
 varfn=~exp(b1-b2)*time*dose*exp(-exp(b1)*time),
 par=0.1, pvar=list(b1=log(0.5),b2=log(0.2)), pell=5,
 distribution="Student t")
# multivariate Laplace distribution
elliptic(conc, model=~exp(absorption-volume)*
 dose/(exp(absorption)-exp(elimination))*
 (exp(-exp(elimination)*time)-exp(-exp(absorption)*time)),
```

```
 preg=list(absorption=log(0.5),elimination=log(0.4),volume=log(0.1)),
 varfn=~exp(b1-b2)*time*dose*exp(-exp(b1)*time),
 par=0.1, pvar=list(b1=log(0.5),b2=log(0.2)), pell=5,
 distribution="Laplace")
#
# generalized logistic regression with square-root transformation
# and square  link
time <- rep(seq(10,200,by=10),2)
mu <- function(p) {
 yinf <- exp(p[2])
 yinf*(1+((yinf/exp(p[1]))^p[4]-1)*exp(-yinf^p[4]
  *exp(p[3])*time))^(-1/p[4])}
y <- matrix(rnorm(40,sqrt(mu(c(2,1.5,0.05,-2))),0.05)^2,ncol=20,byrow=TRUE)
y[,2:20] <- y[,2:20]+0.5*(y[,1:19]-matrix(mu(c(2,1.5,0.05,-2)),
 ncol=20,byrow=TRUE)[,1:19])
y <- ifelse(y>0,y,0.01)
# with builtin function
# independence
elliptic(y, model="logistic", preg=c(2,1,0.1,-1), trans="sqrt",
 link="square")
# the same model with AR(1)
elliptic(y, model="logistic", preg=c(2,1,0.1,-1), trans="sqrt",
 link="square", par=0.4)
# the same model with AR(1) and one component of variance
elliptic(y, model="logistic", preg=c(2,1,0.1,-1),
 trans="sqrt", link="square", pre=1, par=0.4)
# or equivalently with user-specified function
# independence
elliptic(y, model=mu, preg=c(2,1,0.1,-1), trans="sqrt",
 link="square")
# the same model with AR(1)
elliptic(y, model=mu, preg=c(2,1,0.1,-1), trans="sqrt",
 link="square", par=0.4)
# the same model with AR(1) and one component of variance
elliptic(y, model=mu, preg=c(2,1,0.1,-1),
 trans="sqrt", link="square", pre=1, par=0.4)
# or equivalently with user-specified formula
# independence
elliptic(y, model=~exp(yinf)*(1+((exp(yinf-y0))^b4-1)*
 exp(-exp(yinf*b4+b3)*time))^(-1/b4),
 preg=list(y0=2,yinf=1,b3=0.1,b4=-1), trans="sqrt", link="square")
# the same model with AR(1)
elliptic(y, model=~exp(yinf)*(1+((exp(yinf-y0))^b4-1)*
 exp(-exp(yinf*b4+b3)*time))^(-1/b4),
 preg=list(y0=2,yinf=1,b3=0.1,b4=-1), trans="sqrt",
 link="square", par=0.1)
# add one component of variance
elliptic(y, model=~exp(yinf)*(1+((exp(yinf-y0))^b4-1)*
 exp(-exp(yinf*b4+b3)*time))^(-1/b4),
 preg=list(y0=2,yinf=1,b3=0.1,b4=-1),
 trans="sqrt", link="square", pre=1, par=0.1)
#
# multivariate power exponential and Student t distributions for outliers
```

```
y <- matrix(rcauchy(40,mu(c(2,1.5,0.05,-2)),0.05),ncol=20,byrow=TRUE)
y[,2:20] <- y[,2:20]+0.5*(y[,1:19]-matrix(mu(c(2,1.5,0.05,-2)),
 ncol=20,byrow=TRUE)[,1:19])
y <- ifelse(y>0,y,0.01)
# first with normal distribution
elliptic(y, model="logistic", preg=c(1,1,0.1,-1))
elliptic(y, model="logistic", preg=c(1,1,0.1,-1), par=0.5)
# then power exponential
elliptic(y, model="logistic", preg=c(1,1,0.1,-1), pell=1,
 distribution="power exponential")
elliptic(y, model="logistic", preg=c(1,1,0.1,-1), par=0.5, pell=1,
 distribution="power exponential")
# finally Student t
elliptic(y, model="logistic", preg=c(1,1,0.1,-1), pell=1,
 distribution="Student t")
elliptic(y, model="logistic", preg=c(1,1,0.1,-1), par=0.5, pell=1,
 distribution="Student t")
```

---

pergram *Calculate and Plot a Periodogram*

---

## Description

pergram calculates the values of a periodogram, plot.pergram plots it, and plot.cum.pergram plots the corresponding cumulative periodogram.

## Usage

```
pergram(y)

## S3 method for class 'pergram'
plot(x, add = FALSE, lty = 1, xlab = "Frequency",
  ylab = "Periodogram", main = "Periodogram", ylim = c(0, max(po[,
  2])), ...)

## S3 method for class 'pergram'
plot_cum(x, xlab = "Frequency", ylab = "Periodogram",
  main = "Cumulative periodogram", ylim = c(0, max(cpo + 1.358/(a +
  0.12 + 0.11/a))), ...)
```

## Arguments

| | |
|---|---|
| y | A time series vector. |
| x | Plotting parameters |
| add | If TRUE, adds a new periodogram to an existing plot. |
| lty | Plotting parameters |

| | |
|---|---|
| xlab | Plotting parameters |
| ylab | Plotting parameters |
| main | Plotting parameters |
| ylim | Plotting parameters |
| ... | Plotting parameters |

## Value

pergram prints and returns a two-column matrix of class, pergram, containing the periodogram.

## Methods (by generic)

- plot: Plot method
- plot_cum: Plot_cum method

## Author(s)

J.K. Lindsey

## Examples

```
y <- rnorm(100)
print(z <- pergram(y))
plot(z)
plot_cum(z)
```

---

# potthoff *Potthoff and Roy Growth Curve Model*

---

## Description

potthoff fits the Potthoff and Roy repeated measurements growth curve model with unstructured covariance matrix to completely balanced data.

## Usage

```
potthoff(response, x = NULL, ccov = NULL, times = NULL, torder = 0,
  orthogonal = TRUE)
```

### Arguments

| | |
|---|---|
| `response` | A matrix or dataframe of response values. |
| `x` | A matrix defining the complete intersubject differences or a Wilkinson and Rogers formula that will create one. |
| `ccov` | A matrix of columns of the baseline covariates to be actually fitted, with one row per individual or a W&R formula that will create one. |
| `times` | A vector of unequally spaced times when they are the same for all individuals. Not necessary if equally spaced. |
| `torder` | Order of the polynomial in time to be fitted. If non-numeric, the full model in time is fitted. |
| `orthogonal` | If TRUE, uses orthogonal polynomials for time, otherwise only centres times at their mean. |

### Value

A list of class potthoff is returned.

### Author(s)

J.K. Lindsey

### See Also

`carma`, `elliptic`, `lm`.

### Examples

```
y <- matrix(rnorm(40),ncol=5)
x <- gl(2,4)
# full model with treatment effect
potthoff(y, ~x, torder="f", ccov=~x)
# no time trend with treatment effect
potthoff(y, ~x, torder=0, ccov=~x)
# quadratic time with treatment effect
potthoff(y, ~x, torder=2, ccov=~x)
# full model without treatment effect
potthoff(y, ~x, torder="f")
# linear time without treatment effect
potthoff(y, ~x, torder=1)
```

rmaov *Split-plot ANOVA Model*

## Description

rmaov performs the classical balanced split-plot ANOVA, with summary providing the table. This is the so-called repeated measures ANOVA.

## Usage

```
rmaov(response, tvcov = NULL, ccov = NULL, analysis = TRUE)
```

## Arguments

| | |
|---|---|
| response | A matrix or dataframe of response values with units as rows and repeated measures as columns. |
| tvcov | A numeric vector or factor variable defining the clusters. If there are several levels of nesting, a matrix or dataframe with columns of such variables defining the nested clusters starting with the highest level (that is, from slowest to fastest varying). If not provided, each response value of a unit is assumed to belong to a different cluster (that is, one factor with ncol(response) levels is assumed). |
| ccov | A vector or factor variable for one inter-subject covariate or a matrix, dataframe, or list of several such variables. |
| analysis | If FALSE, the design matrix is set up, but the analysis is not performed. |

## Details

For unbalanced data, elliptic will perform the analysis for one or two levels of nesting.

## Value

The fitted model is returned.

## Author(s)

Ralf Goertz (ralf.goertz@uni-jena.de)

## See Also

carma, elliptic, lm, potthoff.

## Examples

```
# vision data for 7 individuals, with response a 7x8 matrix
# two levels of nesting: 4 levels of power for each eye
y <- matrix(rnorm(56),ncol=8)
tvc <- data.frame(eye=c(rep(1,4),rep(2,4)),power=c(1:4,1:4))
summary(rmaov(y, tvc))
```

# Index