

Package ‘aster’

April 17, 2009

Version 0.7-7

Date 2009-03-23

Title Aster Models

Author Charles J. Geyer <charlie@stat.umn.edu>.

Maintainer Charles J. Geyer <charlie@stat.umn.edu>

Depends R (>= 2.4.1), trust

Description functions and datasets for Aster modeling (forest graph exponential family conditional or unconditional canonical statistic models for life history analysis)

License X11

URL <http://www.stat.umn.edu/geyer/aster/>

Repository CRAN

Date/Publication 2009-03-24 07:58:07

R topics documented:

anova.aster	2
aphid	3
aster	4
chamae	8
chamae2	9
echin2	11
echinacea	12
families	13
mlogl	16
predict.aster	17
raster	21
sim	23
summary.aster	24
truncated	25

Index	27
--------------	-----------

 anova.aster

Analysis of Deviance for Aster Model Fits

Description

Compute an analysis of deviance table for two or more aster model fits.

Usage

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

Arguments

`object, ...` objects of class "aster", typically the result of a call to [aster](#), or a list of objects of class "aster" for the "asterlist" method.

Details

Constructs a table having a row for the degrees of freedom and deviance for each model. For all but the first model, the change in degrees of freedom and deviance is also given, as is the corresponding asymptotic P value.

Value

An object of class "anova" inheriting from class "data.frame".

Warning

The comparison between two or more models by `anova` or `anova.asterlist` will only be valid if they are (1) fitted to the same dataset, (2) models are nested, (3) models are of the same type (all conditional or all unconditional), (4) have the same dependence graph and exponential families. None of this is currently checked.

See Also

[aster](#), [anova](#).

Examples

```
### see package vignette for explanation ###
library(aster)
data(echinacea)
vars <- c("ld02", "ld03", "ld04", "f102", "f103", "f104",
          "hdct02", "hdct03", "hdct04")
redata <- reshape(echinacea, varying = list(vars), direction = "long",
                  timevar = "varb", times = as.factor(vars), v.names = "resp")
redata <- data.frame(redata, root = 1)
pred <- c(0, 1, 2, 1, 2, 3, 4, 5, 6)
```

```
fam <- c(1, 1, 1, 1, 1, 1, 3, 3, 3)
hdct <- grep("hdct", as.character(redata$varb))
hdct <- is.element(seq(along = redata$varb), hdct)
redata <- data.frame(redata, hdct = as.integer(hdct))
aout3 <- aster(resp ~ varb + nsloc + ewloc + pop * hdct,
  pred, fam, varb, id, root, data = redata)
aout4 <- aster(resp ~ varb + nsloc + ewloc + pop * hdct - pop,
  pred, fam, varb, id, root, data = redata)
anova(aout4, aout3)
```

aphid

*Life History Data on Uroleucon rudbeckiae***Description**

Data on life history traits for the brown ambrosia aphid *Uroleucon rudbeckiae*

Usage

```
aphid
```

Format

A data frame with records for 18 insects. Data are already in “long” format; no need to reshape.

resp Response vector.

varb Categorical. Gives node of graphical model corresponding to each component of `resp`. See details below.

root All ones. Root variables for graphical model.

id Categorical. Indicates individual plants.

Details

The levels of `varb` indicate nodes of the graphical model to which the corresponding elements of the response vector `resp` belong. This is the typical “long” format produced by the R `reshape` function. For each individual, there are several response variables. All response variables are combined in one vector `resp`. The variable `varb` indicates which “original” variable the number was for. The variable `id` indicates which individual the number was for. The levels of `varb`, which are the names of the “original” variables are the following. `S1` through `S13` are Bernoulli: one if alive, zero if dead. `B2` through `B9` are conditionally Poisson: the number of offspring in the corresponding time period. Some variables in the original data that were zero have been deleted.

References

These data were published in the following, where they were analyzed by non-aster methods.

Lenski, R.-E. and Service, P.-M. (1982). The statistical analysis of population growth rates calculated from schedules of survivorship and fecundity. *Ecology*, **63**, 655-662.

Examples

```
data(aphid)
### wide version
aphidw <- reshape(aphid, direction = "wide", timevar = "varb",
  v.names = "resp", varying = list(levels(aphid$varb)))
```

 aster

Aster Models

Description

Fits Aster Models.

Usage

```
aster(x, ...)

## Default S3 method:
aster(x, root, pred, fam, modmat, parm,
  type = c("unconditional", "conditional"), famlist = fam.default(),
  origin, origin.type = c("model.type", "unconditional", "conditional"),
  method = c("trust", "nlm", "CG", "L-BFGS-B"), fscale, maxiter = 1000,
  nowarn = TRUE, newton = TRUE, optout = FALSE, coef.names, ...)

## S3 method for class 'formula':
aster(formula, pred, fam, varvar, idvar, root,
  data, parm, type = c("unconditional", "conditional"),
  famlist = fam.default(),
  origin, origin.type = c("model.type", "unconditional", "conditional"),
  method = c("trust", "nlm", "CG", "L-BFGS-B"), fscale, maxiter = 1000,
  nowarn = TRUE, newton = TRUE, optout = FALSE, ...)
```

Arguments

x	<p>an <code>nind</code> by <code>nnode</code> matrix, the data for an aster model. The rows are independent and identically modeled random vectors. See details below for further requirements.</p> <p><code>aster.formula</code> constructs such an <code>x</code> from the response in its formula. Hence data for <code>aster.formula</code> must have <code>nind * nnode</code> rows.</p>
root	<p>an object of the same shape as <code>x</code>, the root data. For <code>aster.default</code> an <code>nind</code> by <code>nnode</code> matrix, For <code>aster.formula</code> an <code>nind * nnode</code> vector.</p>
pred	<p>an integer vector of length <code>nnode</code> determining the dependence graph of the aster model. <code>pred[j]</code> is the index of the predecessor of the node with index <code>j</code> unless the predecessor is a root node, in which case <code>pred[j] == 0</code>. See details below for further requirements.</p>

<code>fam</code>	an integer vector of length <code>nnode</code> determining the exponential family structure of the aster model. Each element is an index into the vector of family specifications given by the argument <code>famlist</code> .
<code>modmat</code>	an <code>nind</code> by <code>nnode</code> by <code>ncoef</code> three-dimensional array, the model matrix. <code>aster.formula</code> constructs such a <code>modmat</code> from its formula, the data frame <code>data</code> , and the variables in the environment of the formula.
<code>parm</code>	usually missing. Otherwise a vector of length <code>ncoef</code> giving a starting point for the optimization.
<code>type</code>	type of model. The value of this argument can be abbreviated.
<code>famlist</code>	a list of family specifications (see families).
<code>origin</code>	Distinguished point in parameter space. May be missing, in which case an unspecified default is provided. See details below for further explanation.
<code>origin.type</code>	Parameter space in which specified distinguished point is located. If "conditional" then argument "origin" is a conditional canonical parameter value. If "unconditional" then argument "origin" is an unconditional canonical parameter value. If "model.type" then the type is taken from argument "type". The value of this argument can be abbreviated.
<code>method</code>	optimization method. If "trust" then the <code>trust</code> function is used. If "nlm" then the <code>nlm</code> function is used. Otherwise the <code>optim</code> function is used with the specified <code>method</code> supplied to it. The value of this argument can be abbreviated.
<code>fscale</code>	an estimate of the size of the log likelihood at the maximum. Defaults to <code>nind</code> .
<code>maxiter</code>	maximum number of iterations. Defaults to '1000'.
<code>nowarn</code>	if TRUE (the default), suppress warnings from the optimization routine.
<code>newton</code>	if TRUE (the default), do one Newton iteration on the result produced by the optimization routine, except when <code>method == "trust"</code> when no such Newton iteration is done, regardless of the value of <code>newton</code> , because <code>trust</code> always terminates with a Newton iteration when it converges.
<code>optout</code>	if TRUE, save the entire result of the optimization routine (<code>trust</code> , <code>nlm</code> , or <code>optim</code> , as the case may be).
<code>coef.names</code>	names of the regression coefficients. If missing, <code>dimnames(modmat)[[3]]</code> is used. In <code>aster.formula</code> these are produced automatically by the R formula machinery.
<code>...</code>	other arguments passed to the optimization method.
<code>formula</code>	a symbolic description of the model to be fit. See lm , glm , and formula for discussions of the R formula mini-language.
<code>varvar</code>	a variable of the same length as the response in the formula that is a factor whose levels are character strings treated as variable names. The number of variable names is <code>nnode</code> . Must be of the form <code>rep(vars, each = nind)</code> where <code>vars</code> is a vector of variable names. Usually found in the data frame <code>data</code> when this is produced by the reshape function.
<code>idvar</code>	a variable of the same length as the response in the formula that indexes individuals. The number of individuals is <code>nind</code> . Must be of the form <code>rep(inds, times = nnode)</code> where <code>inds</code> is a vector of labels for individuals. Usually found in the data frame <code>data</code> when this is produced by the reshape function.

`data` an optional 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 `aster` is called. Usually produced by the `reshape` function.

Details

The vector `pred` must satisfy all (`pred < seq(along = pred)`), that is, each predecessor must precede in the order given in `pred`. The vector `pred` defines a function p .

The joint distribution of the data matrix x is a product of conditionals

$$\prod_{i \in I} \prod_{j \in J} \Pr\{X_{ij} | X_{ip(j)}\}$$

When $p(j) = 0$, the notation $X_{ip(j)}$ means `root[i, j]`. Other elements of the matrix `root` are not used.

The conditional distribution $\Pr\{X_{ij} | X_{ip(j)}\}$ is the $X_{ip(j)}$ -fold convolution of the j -th family in the vector `fam`, a one-parameter exponential family (i.e., the sum of $X_{ip(j)}$ i.i.d. terms having this one-parameter exponential family distribution).

For `type == "conditional"` the canonical parameter vector θ_{ij} is modeled in GLM fashion as $\theta = a + M\beta$ where M is the model matrix `modmat` and a is the distinguished point `origin`. Since the “vector” θ is actually a matrix, the “matrix” M must correspondingly be a three-dimensional array. So $\theta = a + M\beta$ written out in full is

$$\theta_{ij} = a_{ij} + \sum_{k \in K} m_{ijk} \beta_k$$

This specifies the log likelihood.

For `type == "unconditional"` the canonical parameter vector for an unconditional model is modeled in GLM fashion as $\varphi = a + M\beta$ (where the notation is as above). The unconditional canonical parameters are then specified in terms of the conditional ones by

$$\varphi_{ij} = \theta_{ij} - \sum_{k \in S(j)} \psi_k(\theta_{ik})$$

where $S(j)$ denotes the set of successors of j , the k such that $p(k) = j$, and ψ_k is the cumulant function for the k -th exponential family. This rather crazy looking formulation is an invertible change of parameter and makes φ the canonical parameter and x the canonical statistic of a full flat unconditional exponential family. Again, this specifies the log likelihood.

In versions of `aster` prior to version 0.6 there was no a in the model specification, which is the same as specifying $a = 0$ in the current specification. If a is in the column space of the model matrix, that is, if there exists an α such that $a = M\alpha$, then there is no difference in the model specified with a and the one with $a = 0$. The maximum likelihood regression coefficients $\hat{\beta}$ will be different, but the maximum likelihood estimates of all other parameters (conditional and unconditional, canonical and mean value) will be the same. This is the usual case and explains why “linear” models (with $a = 0$) as opposed to “affine” models (with general a) are popular. In the unusual case where a is not in the column space of the design matrix, then affine models are a generalization of linear models: the two are not equivalent, their maximum likelihood estimates are not the same in any parameterization.

In order to use the R model formula mini-language we must flatten the dimensionality, making the model matrix `modmat` two-dimensional (a true matrix). This must be done as if by `matrix(modmat, ncol = ncoef)`, which imposes the requirements on `varvar` and `idvar` given in the arguments section: they must look like `row(x)` and `col(x)` modulo relabeling. Then `x` and `root` become one-dimensional, done as if by `as.numeric(x)` and `as.numeric(root)`.

The standard way to do this in R is to use the `reshape` function on a data frame in which the columns of the `x` matrix are variables in the data frame. `reshape` automatically puts things in the right order and creates `varvar` and `idvar`.

Value

`aster` returns an object of class inheriting from "aster". `aster.formula`, returns an object of class "aster" and subclass "aster.formula".

The function `summary` (i.e., `summary.aster`) can be used to obtain or print a summary of the results, the function `anova` (i.e., `anova.aster`) to produce an analysis of deviance table, and the function `predict` (i.e., `predict.aster`) to produce predicted values and standard errors.

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

<code>coefficients</code>	a named vector of coefficients.
<code>rank</code>	the numeric rank of the fitted generalized linear model part of the aster model (i.e., the rank of <code>modmat</code>).
<code>deviance</code>	up to a constant, minus twice the maximized log-likelihood.
<code>iter</code>	the number of iterations used by the optimization method.
<code>converged</code>	logical. Was the optimization algorithm judged to have converged?
<code>code</code>	integer. The convergence code returned by the optimization method.
<code>gradient</code>	The gradient vector of minus the log likelihood at the fitted <code>coefficients</code> vector.
<code>hessian</code>	The Hessian matrix of minus the log likelihood (i.e., the observed Fisher information) at the fitted <code>coefficients</code> vector. This is also the expected Fisher information when <code>type == "unconditional"</code> .
<code>fisher</code>	Expected Fisher information at the fitted <code>coefficients</code> vector.
<code>optout</code>	The object returned by the optimization routine (<code>trust</code> , <code>nlm</code> , or <code>optim</code>). Only returned when the argument <code>optout</code> is TRUE.
<code>call</code>	the matched call.
<code>formula</code>	the formula supplied.
<code>terms</code>	the <code>terms</code> object used.
<code>data</code>	the <code>data</code> argument.

References

- Geyer, C. J., Wagenius, S., and Shaw, R. G. (2007) Aster Models for Life History Analysis. *Biometrika* **94** 415–426.
- Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2007) Unifying Life History Analysis for Inference of Fitness and population growth. *American Naturalist* **172** E35–E47. (e-paper <http://www.journals.uchicago.edu/doi/full/10.1086/588063>)

See Also

[anova.aster](#), [summary.aster](#), and [predict.aster](#)

Examples

```
### see package vignette for explanation ###
library(aster)
data(echinacea)
vars <- c("ld02", "ld03", "ld04", "fl02", "fl03", "fl04",
         "hdct02", "hdct03", "hdct04")
redata <- reshape(echinacea, varying = list(vars), direction = "long",
                 timevar = "varb", times = as.factor(vars), v.names = "resp")
redata <- data.frame(redata, root = 1)
pred <- c(0, 1, 2, 1, 2, 3, 4, 5, 6)
fam <- c(1, 1, 1, 1, 1, 1, 3, 3, 3)
hdct <- grep("hdct", as.character(redata$varb))
hdct <- is.element(seq(along = redata$varb), hdct)
redata <- data.frame(redata, hdct = as.integer(hdct))
aout4 <- aster(resp ~ varb + nsloc + ewloc + pop * hdct - pop,
              pred, fam, varb, id, root, data = redata)
summary(aout4, show.graph = TRUE)
```

chamae

Life History Data on Chamaecrista fasciculata

Description

Data on life history traits for the partridge pea *Chamaecrista fasciculata*

Usage

chamae

Format

A data frame with records for 2235 plants. Data are already in “long” format; no need to reshape.

resp Response vector.

varb Categorical. Gives node of graphical model corresponding to each component of *resp*. See details below.

root All ones. Root variables for graphical model.

id Categorical. Indicates individual plants.

STG1N Numerical. Reproductive stage. Integer with only 3 values in this dataset.

LOGLVS Numerical. Log leaf number.

LOGSLA Numerical. Log leaf thickness.

BLK Categorical. Block within experiment.

Details

The levels of `varb` indicate nodes of the graphical model to which the corresponding elements of the response vector `resp` belong. This is the typical “long” format produced by the R `reshape` function. For each individual, there are several response variables. All response variables are combined in one vector `resp`. The variable `varb` indicates which “original” variable the number was for. The variable `id` indicates which individual the number was for. The levels of `varb`, which are the names of the “original” variables are

fecund Fecundity. Bernoulli, One if any fruit, zero if no fruit.

fruit Integer. Number of fruits observed. Greater than or equal 3 if nonzero.

seed Integer. Number of seeds observed from a random sample of 3 of the fruits for this individual.

Source

Julie Etterson http://www.d.umn.edu/biology/faculty/Etterson_page1.htm

References

These data have been previously analyzed by non-aster methods in the following.

Etterson, J.-R. (2004). Evolutionary potential of *Chamaecrista fasciculata* in relation to climate change. I. Clinal patterns of selection along an environmental gradient in the great plains. *Evolution*, **58**, 1446-1458.

Etterson, J.-R., and Shaw, R.-G. (2001). Constraint to adaptive evolution in response to global warming. *Science*, **294**, 151-154.

Examples

```
data(chamae)
### wide version
chamaew <- reshape(chamae, direction = "wide", timevar = "varb",
  v.names = "resp", varying = list(levels(chamae$varb)))
```

chamae2

Life History Data on Chamaecrista fasciculata

Description

Data on life history traits for the partridge pea *Chamaecrista fasciculata*

Usage

chamae2

Format

A data frame with records for 2239 plants. Data are already in “long” format; no need to reshape.

resp Response vector.

varb Categorical. Gives node of graphical model corresponding to each component of `resp`. See details below.

root All ones. Root variables for graphical model.

id Categorical. Indicates individual plants.

STG1N Numerical. Reproductive stage. Integer with only 3 values in this dataset.

LOGLVS Numerical. Log leaf number.

LOGSLA Numerical. Log leaf thickness.

BLK Categorical. Block within experiment.

Details

The levels of `varb` indicate nodes of the graphical model to which the corresponding elements of the response vector `resp` belong. This is the typical “long” format produced by the R `reshape` function. For each individual, there are several response variables. All response variables are combined in one vector `resp`. The variable `varb` indicates which “original” variable the number was for. The variable `id` indicates which individual the number was for. The levels of `varb`, which are the names of the “original” variables are

fecund Fecundity. Bernoulli, One if any fruit, zero if no fruit.

fruit Integer. Number of fruits observed.

Source

Julie Etterson http://www.d.umn.edu/biology/faculty/Etterson_page1.htm

References

These data have been previously analyzed by non-aster methods in the following.

Etterson, J.~R. (2004). Evolutionary potential of *Chamaecrista fasciculata* in relation to climate change. I. Clinal patterns of selection along an environmental gradient in the great plains. *Evolution*, **58**, 1446-1458.

Etterson, J.~R., and Shaw, R.~G. (2001). Constraint to adaptive evolution in response to global warming. *Science*, **294**, 151-154.

Examples

```
data(chamae2)
### wide version
chamae2w <- reshape(chamae2, direction = "wide", timevar = "varb",
  v.names = "resp", varying = list(levels(chamae2$varb)))
```

 echin2

Life History Data on Echinacea angustifolia

Description

Data on life history traits for the purple coneflower *Echinacea angustifolia*

Usage

```
echin2
```

Format

A data frame with records for 557 plants observed over five years. Data are already in “long” format; no need to reshape.

resp Response vector.

varb Categorical. Gives node of graphical model corresponding to each component of `resp`. See details below.

root All ones. Root variables for graphical model.

id Categorical. Indicates individual plants.

flat Categorical. Position in growth chamber.

row Categorical. Row in the field.

posi Numerical. Position within row in the field.

crosstype Categorical. See details.

yearcross Categorical. Year in which cross was done.

Details

The levels of `varb` indicate nodes of the graphical model to which the corresponding elements of the response vector `resp` belong. This is the typical “long” format produced by the R `reshape` function. For each individual, there are several response variables. All response variables are combined in one vector `resp`. The variable `varb` indicates which “original” variable the number was for. The variable `id` indicates which individual the number was for. The levels of `varb`, which are the names of the “original” variables are

lds1 Survival for the first month in the growth chamber.

lds2 Ditto for 2nd month in the growth chamber.

lds3 Ditto for 3rd month in the growth chamber.

ld01 Survival for first year in the field.

ld02 Ditto for 2nd year in the field.

ld03 Ditto for 3rd year in the field.

ld04 Ditto for 4th year in the field.

ld05 Ditto for 5th year in the field.

roct2003 Rosette count, measure of size and vigor, recorded for 3rd year in the field.

roct2004 Ditto for 4th year in the field.

roct2005 Ditto for 5th year in the field.

These data are complicated by the experiment being done in two parts. Plants start their life indoors in a growth chamber. The predictor variable `flat` only makes sense during this time in which three response variables `lds1`, `lds2`, and `lds3` are observed. After three months in the growth chamber, the plants (if they survived, i. e., if `lds3 == 1`) were planted in an experimental field plot outdoors. The variables `row` and `posi` only make sense during this time in which all of the rest of the response variables are observed. Because of certain predictor variables only making sense with respect to certain components of the response vector, the R formula mini-language is unable to cope, and model matrices must be constructed "by hand".

Echinacea angustifolia is native to North American tallgrass prairie, which was once extensive but now exists only in isolated remnants. To evaluate the effects of different mating regimes on the fitness of resulting progeny, crosses were conducted to produce progeny of (a) mates from different remnants, (b) mates chosen at random from the same remnant, and (c) mates known to share maternal parent. These three categories are the three levels of `crosstype`.

Source

Stuart Wagenius, <http://www.chicagobotanic.org/research/staff/wagenius.php>

Examples

```
data(echin2)
```

echinacea

Life History Data on Echinacea angustifolia

Description

Data on life history traits for the purple coneflower *Echinacea angustifolia*

Usage

```
echinacea
```

Format

A data frame with records for 570 plants observed over three years.

ld02 Indicator of being alive in 2002.

ld03 Ditto for 2003.

ld04 Ditto for 2004.

fl02 Indicator of flowering 2002.
fl03 Ditto for 2003.
fl04 Ditto for 2004.
hdct02 Count of number of flower heads in 2002.
hdct03 Ditto for 2003.
hdct04 Ditto for 2004.
pop the remnant population of origin of the plant (all plants were grown together, pop encodes ancestry).
ewloc east-west location in plot.
nsloc north-south location in plot.

Source

Stuart Wagenius, <http://www.chicagobotanic.org/research/staff/wagenius.php>

Examples

```
library(aster)
data(echinacea)
vars <- c("ld02", "ld03", "ld04", "fl02", "fl03", "fl04",
          "hdct02", "hdct03", "hdct04")
redata <- reshape(echinacea, varying = list(vars), direction = "long",
                  timevar = "varb", times = as.factor(vars), v.names = "resp")
names(echinacea)
dim(echinacea)
names(redata)
dim(redata)
```

families

Families for Aster Models

Description

Families (response models) known to the package. These functions construct simple family specifications used in specifying models for `aster` and `mlogl`. They are mostly for convenience, since the specifications are easy to construct by hand.

Usage

```
fam.bernoulli()
fam.poisson()
fam.truncated.poisson(truncation)
fam.negative.binomial(size)
fam.truncated.negative.binomial(size, truncation)
fam.normal.location(sd)
fam.default()
famfun(fam, deriv, theta)
```

Arguments

<code>truncation</code>	the truncation point, called k in the details section below.
<code>size</code>	the sample size. May be non-integer.
<code>sd</code>	the standard deviation. May be non-integer.
<code>fam</code>	a family specification, which is a list of class <code>"astfam"</code> containing, at least one element named <code>"name"</code> and perhaps other elements specifying hyperparameters.
<code>deriv</code>	derivative wanted: 0, 1, or 2.
<code>theta</code>	value of the canonical parameter.

Details

Currently implemented families are

"bernoulli" Bernoulli. The mean value parameter μ is the success probability. The canonical parameter is $\theta = \log(\mu) - \log(1 - \mu)$, also called logit of μ . The first derivative of the cumulant function has the value μ and the second derivative of the cumulant function has the value $\mu(1 - \mu)$.

"poisson" Poisson. The mean value parameter μ is the mean of the Poisson distribution. The canonical parameter is $\theta = \log(\mu)$. The first and second derivatives of the cumulant function both have the value μ .

"truncated.poisson" Poisson conditioned on being strictly greater than k , specified by the argument `truncation`. Let μ be the mean of the corresponding untruncated Poisson distribution. Then the canonical parameters for both truncated and untruncated distributions are the same $\theta = \log(\mu)$. Let Y be a Poisson random variable having the same mean parameter as this distribution, and define

$$\beta = \frac{\Pr\{Y > k + 1\}}{\Pr\{Y = k + 1\}}$$

Then the mean value parameter and first derivative of the cumulant function of this distribution has the value

$$\tau = \mu + \frac{k + 1}{1 + \beta}$$

and the second derivative of the cumulant function has the value

$$\mu \left[1 - \frac{k + 1}{1 + \beta} \left(1 - \frac{k + 1}{\mu} \cdot \frac{\beta}{1 + \beta} \right) \right]$$

"negative.binomial" Negative binomial. The size parameter α may be noninteger, meaning the cumulant function is α times the cumulant function of the geometric distribution. The mean value parameter μ is the mean of the negative binomial distribution. The success probability parameter is

$$p = \frac{\alpha}{\mu + \alpha}.$$

The canonical parameter is $\theta = \log(1 - p)$. Since $1 - p < 1$, the canonical parameter space is restricted, the set of θ such that $\theta < 0$. This is, however, a regular exponential family (the log likelihood goes to minus infinity as θ converges to the boundary of the parameter space, so the

constraint $\theta < 0$ plays no role in maximum likelihood estimation so long as the optimization software is not too stupid. There will be no problems so long as the default optimizer (`trust`) is used. Since zero is not in the canonical parameter space a negative default origin is used. The first derivative of the cumulant function has the value

$$\mu = \alpha \frac{1-p}{p}$$

and the second derivative has the value

$$\alpha \frac{1-p}{p^2}.$$

"truncated.negative.binomial" Negative binomial conditioned on being strictly greater than k , specified by the argument `truncation`. Let p be the success probability parameter of the corresponding untruncated negative binomial distribution. Then the canonical parameters for both truncated and untruncated distributions are the same $\theta = \log(1-p)$, and consequently the canonical parameter spaces are the same, the set of θ such that $\theta < 0$, and both models are regular exponential families. Let Y be an untruncated negative binomial random variable having the same size and success probability parameters as this distribution. and define

$$\beta = \frac{\Pr\{Y > k + 1\}}{\Pr\{Y = k + 1\}}$$

Then the mean value parameter and first derivative of the cumulant function of this distribution has the value

$$\tau = \mu + \frac{k+1}{p(1+\beta)}$$

and the second derivative is too complicated to write here (the formula can be found in the vignette `trunc.pdf`).

"normal.location" Normal, unknown mean, known variance. The `sd` (standard deviation) parameter σ may be noninteger, meaning the cumulant function is σ^2 times the cumulant function of the standard normal distribution. The mean value parameter μ is the mean of the normal distribution. The canonical parameter is $\theta = \mu/\sigma^2$. The first derivative of the cumulant function has the value

$$\mu = \sigma^2 \theta$$

and the second derivative has the value

$$\sigma^2.$$

Value

For all but `fam.default`, a list of class `"astfam"` giving name and values of any hyperparameters. For `fam.default`, a list each element of which is of class `"astfam"`. The list of families which were hard coded in earlier versions of the package.

See Also

[aster](#) and [mlogl](#)

Examples

```
### mean of poisson with mean 0.2
famfun(fam.poisson(), 1, log(0.2))
### variance of poisson with mean 0.2
famfun(fam.poisson(), 2, log(0.2))
### mean of poisson with mean 0.2 conditioned on being nonzero
famfun(fam.truncated.poisson(trunc = 0), 1, log(0.2))
### variance of poisson with mean 0.2 conditioned on being nonzero
famfun(fam.truncated.poisson(trunc = 0), 2, log(0.2))
```

mlogl

Minus Log Likelihood for Aster Models

Description

Minus the Log Likelihood for an Aster model, and its first and second derivative. This function is called inside `aster`. Users generally do not need to call it directly.

Usage

```
mlogl(parm, pred, fam, x, root, modmat, deriv = 0,
      type = c("unconditional", "conditional"), famlist = fam.default(),
      origin, origin.type = c("model.type", "unconditional", "conditional"))
```

Arguments

parm	parameter value (vector of regression coefficients) where we evaluate the log likelihood, etc. We also refer to <code>length(parm)</code> as <code>ncoef</code> .
pred	integer vector determining the graph. <code>pred[j]</code> is the index of the predecessor of the node with index <code>j</code> unless the predecessor is a root node, in which case <code>pred[j] == 0</code> . We also refer to <code>length(pred)</code> as <code>nnode</code> .
fam	an integer vector of length <code>nnode</code> determining the exponential family structure of the aster model. Each element is an index into the vector of family specifications given by the argument <code>famlist</code> .
x	the response. If a matrix, rows are individuals, and columns are variables (nodes of graphical model). So <code>ncol(x) == nnode</code> and we also refer to <code>nrow(x)</code> as <code>nind</code> . If not a matrix, then <code>x</code> must be as if it were such a matrix and then dimension information removed by <code>x = as.numeric(x)</code> .
root	A matrix or vector like <code>x</code> . Data <code>root[i, j]</code> is the data for the founder that is the predecessor of the response <code>x[i, j]</code> and is ignored when <code>pred[j] > 0</code> .
modmat	a three-dimensional array, <code>nind</code> by <code>nnode</code> by <code>ncoef</code> , the model matrix. Or a matrix, <code>nind * nnode</code> by <code>ncoef</code> (when <code>x</code> and <code>root</code> are one-dimensional of length <code>nind * nnode</code>).
deriv	derivative wanted: 0, 1, or 2.

type	type of model. The value of this argument can be abbreviated.
famlist	a list of family specifications (see families).
origin	Distinguished point in parameter space. May be missing, in which case an unspecified default is provided. See aster for further explanation.
origin.type	Parameter space in which specified distinguished point is located. If "conditional" then argument "origin" is a conditional canonical parameter value. If "unconditional" then argument "origin" is an unconditional canonical parameter value. If "model.type" then the type is taken from argument "type". The value of this argument can be abbreviated.

Value

a list containing some of the following components:

value	minus the log likelihood.
gradient	minus the first derivative vector of the log likelihood (minus the score).
hessian	minus the second derivative matrix of the log likelihood (observed Fisher information).

predict.aster	<i>Predict Method for Aster Model Fits</i>
---------------	--

Description

Obtains predictions and optionally estimates standard errors of those predictions from a fitted Aster model object.

Usage

```
## S3 method for class 'aster':
predict(object, x, root, modmat, amat,
        parm.type = c("mean.value", "canonical"),
        model.type = c("unconditional", "conditional"),
        se.fit = FALSE, info = c("expected", "observed"),
        info.tol = sqrt(.Machine$double.eps), newcoef = NULL, ...)

## S3 method for class 'aster.formula':
predict(object, newdata, varvar, idvar, root, amat,
        parm.type = c("mean.value", "canonical"),
        model.type = c("unconditional", "conditional"),
        se.fit = FALSE, info = c("expected", "observed"),
        info.tol = sqrt(.Machine$double.eps), newcoef = NULL, ...)
```

Arguments

object	a fitted object of class inheriting from "aster" or "aster.formula".
modmat	<p>a model matrix to use instead of <code>object\$modmat</code>. Must have the same structure (three-dimensional array, first index runs over individuals, second over nodes of the graphical model, third over covariates. Must have the same second and third dimensions as <code>object\$modmat</code>. The second and third components of <code>dimnames(modmat)</code> and <code>dimnames(object\$modmat)</code> must also be the same.</p> <p>May be missing, in which case <code>object\$modmat</code> is used.</p> <p><code>predict.aster.formula</code> constructs such a <code>modmat</code> from <code>object\$formula</code>, the data frame <code>newdata</code>, and the variables in the environment of the formula. When <code>newdata</code> is missing, then <code>object\$modmat</code> is used.</p>
x	<p>response. Ignored and may be missing unless <code>parm.type == "mean.value" && model.type == "conditional"</code>. Even then may be missing when <code>modmat</code> is missing, in which case <code>object\$x</code> is used. A matrix whose first and second dimensions and the corresponding <code>dimnames</code> agrees with those of <code>modmat</code> and <code>object\$modmat</code>.</p> <p><code>predict.aster.formula</code> constructs such an <code>x</code> from the response variable name in <code>object\$formula</code>, the data frame <code>newdata</code>, and the variables in the environment of the formula. When <code>newdata</code> is missing, then <code>object\$x</code> is used.</p>
root	<p>root data. Ignored and may be missing unless <code>parm.type == "mean.value"</code>. Even then may be missing when <code>modmat</code> is missing, in which case <code>object\$root</code> is used. A matrix of the same form as <code>x</code>.</p> <p><code>predict.aster.formula</code> looks up the variable supplied as the argument <code>root</code> in the data frame <code>newdata</code> or in the variables in the environment of the formula and makes it a matrix of the same form as <code>x</code>. When <code>newdata</code> is missing, then <code>object\$root</code> is used.</p>
amat	<p>if <code>zeta</code> is the requested prediction (mean value or canonical, unconditional or conditional, depending on <code>parm.type</code> and <code>model.type</code>), then we predict the linear function <code>t(amat) %*% zeta</code>. May be missing, in which case the identity linear function is used.</p> <p>For <code>predict.aster</code>, a three-dimensional array with <code>dim(amat)[1:2] == dim(modmat)[1:2]</code>.</p> <p>For <code>predict.aster.formula</code>, a three-dimensional array of the same dimensions as required for <code>predict.aster</code> (even though <code>modmat</code> is not provided). First dimension is number of individuals in <code>newdata</code>, if provided, otherwise number of individuals in <code>object\$data</code>. Second dimension is number of variables (<code>length(object\$pred)</code>).</p>
parm.type	the type of parameter to predict. The default is mean value parameters (the opposite of the default for <code>predict.glm</code>), the expected value of a linear function of the response under the MLE probability model (also called the MLE of the mean value parameter). The expectation is unconditional or conditional depending on <code>parm.type</code> .

	The alternative "canonical" is the value of a linear function of the MLE of canonical parameters under the MLE probability model. The canonical parameter is unconditional or conditional depending on <code>parm.type</code> . The value of this argument can be abbreviated.
<code>model.type</code>	the type of model in which to predict. The default is "unconditional" in which case the parameters (either mean value or canonical, depending on the value of <code>parm.type</code>) are those of an unconditional model. The alternative is "conditional" in which case the parameters are those of a conditional model. The value of this argument can be abbreviated.
<code>se.fit</code>	logical switch indicating if standard errors are required.
<code>info</code>	the type of Fisher information use to compute standard errors.
<code>info.tol</code>	tolerance for eigenvalues of Fisher information. If <code>eval</code> is the vector of eigenvalues of the information matrix, then <code>eval < cond.tol * max(eval)</code> are considered zero. Hence the corresponding eigenvectors are directions of constancy or recession of the log likelihood.
<code>newdata</code>	optionally, a data frame in which to look for variables with which to predict. If omitted, see <code>modmat</code> above. See also details section below.
<code>varvar</code>	a variable of length <code>nrow(newdata)</code> , typically a variable in <code>newdata</code> that is a factor whose levels are character strings treated as variable names. The number of variable names is <code>nnode</code> . Must be of the form <code>rep(vars, each = nind)</code> where <code>vars</code> is a vector of variable names. Not used if <code>newdata</code> is missing.
<code>idvar</code>	a variable of length <code>nrow(newdata)</code> , typically a variable in <code>newdata</code> that indexes individuals. The number of individuals is <code>nind</code> . Must be of the form <code>rep(inds, times = nnode)</code> where <code>inds</code> is a vector of labels for individuals. Not used if <code>newdata</code> is missing.
<code>newcoef</code>	if not NULL, a variable of length <code>object\$coefficients</code> and used in its place when one wants predictions at other than the fitted coefficient values.
<code>...</code>	further arguments passed to or from other methods.

Details

Note that `model.type` need have nothing to do with the type of the fitted aster model, which is `object$type`.

Whether the fitted model is conditional or unconditional, one typically wants *unconditional* mean value parameters, because conditional mean value parameters for hypothetical individuals depend on the hypothetical data x , which usually makes no scientific sense.

If one does ask for *conditional* mean value parameters, generally the "data" should satisfy `all(x == 1)` and `all(root == 1)`, so that the mean value parameters are "per unit of predecessor variable", that is we "predict" $\psi''(\theta_{ij})$ rather than this multiplied by $X_{ip(j)}$, where $p(j)$ is the mathematical function defined by the R expression `pred[j]`.

Similarly, if `object$type == "conditional"`, then the conditional canonical parameters are a linear function of the regression coefficients $\theta = M\beta$, where M is the model matrix, but one can predict either θ or the unconditional canonical parameters φ , as selected by `model.type`.

Similarly, if `object$type == "unconditional"`, so $\varphi = M\beta$, one can predict either θ or φ as selected by `model.type`.

The specification of the prediction model is confusing because there are so many possibilities. First the “usual” case. The fit was done using a formula, found in `object$formula`. A data frame `newdata` that has the same variables as `object$data`, the data frame used in the fit, but may have different rows (representing hypothetical individuals) is supplied. But `newdata` must specify *all nodes* of the graphical model for each (hypothetical, new) individual, just like `object$data` did for real observed individuals. Hence `newdata` is typically constructed using `reshape`. See also the details section of `aster`.

In this “usual” case we need `varvar` and `idvar` to tell us what rows of `newdata` correspond to which individuals and nodes (the same role they played in the original fit by `aster`). If we are predicting canonical parameters, then we do not need `root` or `x`. If we are predicting unconditional mean value parameters, then we also need `root` but not `x`. If we are predicting conditional mean value parameters, then we also need both `root` and `x`. In the “usual” case, these are found in `newdata` and not supplied as arguments to `predict`. Moreover, `x` is not named “x” but is the response in `out$formula`.

The next case, `predict(object)` with no other arguments, is often used with linear models (`predict.lm`), but we expect will be little used for `aster` models. As for linear models, this “predicts” the observed data. In this case `modmat`, `x`, and `root` are found in `object` and nothing is supplied as an argument to `predict.aster`, except perhaps `amat` if one wants a function of predictions for the observed data.

The final case, also perhaps little used, is a fail-safe mode for problems in which the R formula language just cannot be bludgeoned into doing what you want. This is the same reason `aster.default` exists. Then a model matrix can be constructed “by hand”, and the function `predict.aster` is used instead of `predict.aster.formula`.

Note that it is possible to use a “constructed by hand” model matrix even if `object` was produced by `aster.formula`. Simply explicitly call `predict.aster` rather than `predict` to override the R method dispatch (which would call `predict.aster.formula` in this case).

Value

If `se.fit = FALSE`, a vector of predictions. If `se.fit = TRUE`, a list with components

<code>fit</code>	Predictions
<code>se.fit</code>	Estimated standard errors

Examples

```
### see package vignette for explanation ###
library(aster)
data(echinacea)
vars <- c("ld02", "ld03", "ld04", "fl02", "fl03", "fl04",
         "hdct02", "hdct03", "hdct04")
redata <- reshape(echinacea, varying = list(vars), direction = "long",
                 timevar = "varb", times = as.factor(vars), v.names = "resp")
redata <- data.frame(redata, root = 1)
pred <- c(0, 1, 2, 1, 2, 3, 4, 5, 6)
fam <- c(1, 1, 1, 1, 1, 1, 3, 3, 3)
```

```

hdct <- grep("hdct", as.character(redata$varb))
hdct <- is.element(seq(along = redata$varb), hdct)
redata <- data.frame(redata, hdct = as.integer(hdct))
aout4 <- aster(resp ~ varb + nsloc + ewloc + pop * hdct - pop,
  pred, fam, varb, id, root, data = redata)
newdata <- data.frame(pop = levels(echinacea$pop))
for (v in vars)
  newdata[[v]] <- 1
newdata$root <- 1
newdata$ewloc <- 0
newdata$nsloc <- 0
renewdata <- reshape(newdata, varying = list(vars),
  direction = "long", timevar = "varb", times = as.factor(vars),
  v.names = "resp")
hdct <- grep("hdct", as.character(renewdata$varb))
hdct <- is.element(seq(along = renewdata$varb), hdct)
renewdata <- data.frame(renewdata, hdct = as.integer(hdct))
nind <- nrow(newdata)
nnode <- length(vars)
amat <- array(0, c(nind, nnode, nind))
for (i in 1:nind)
  amat[i, grep("hdct", vars), i] <- 1
foo <- predict(aout4, varvar = varb, idvar = id, root = root,
  newdata = renewdata, se.fit = TRUE, amat = amat)
bar <- cbind(foo$fit, foo$se.fit)
dimnames(bar) <- list(as.character(newdata$pop), c("Estimate", "Std. Error"))
print(bar)

```

raster

Aster Model Simulation

Description

Random generation of data for Aster models.

Usage

```
raster(theta, pred, fam, root, famlist = fam.default())
```

Arguments

theta	canonical parameter of the conditional model. A matrix, rows represent individuals and columns represent nodes in the graphical model.
pred	integer vector of length <code>ncol(theta)</code> determining the graph. <code>pred[j]</code> is the index of the predecessor of the node with index <code>j</code> unless the predecessor is a root node, in which case <code>pred[j] == 0</code> .
fam	integer vector of length <code>ncol(theta)</code> determining the exponential family structure of the aster model. Each element is an index into the vector of family specifications given by the argument <code>famlist</code> .

`root` A matrix of the same dimensions as `theta`. Data `root[i, j]` is the data for the founder that is the predecessor of the `[i, j]` node.

`famlist` a list of family specifications (see [families](#)).

Value

A matrix of the same dimensions as `theta`. The random data for an aster model with the specified graph, parameters, and root data.

See Also

[aster](#)

Examples

```
### see package vignette for explanation ###
data(echinacea)
vars <- c("ld02", "ld03", "ld04", "fl02", "fl03", "fl04",
         "hdct02", "hdct03", "hdct04")
redata <- reshape(echinacea, varying = list(vars),
                 direction = "long", timevar = "varb", times = as.factor(vars),
                 v.names = "resp")
redata <- data.frame(redata, root = 1)
pred <- c(0, 1, 2, 1, 2, 3, 4, 5, 6)
fam <- c(1, 1, 1, 1, 1, 1, 3, 3, 3)
hdct <- grep("hdct", as.character(redata$varb))
hdct <- is.element(seq(along = redata$varb), hdct)
redata <- data.frame(redata, hdct = as.integer(hdct))
aout4 <- aster(resp ~ varb + nsloc + ewloc + pop * hdct - pop,
             pred, fam, varb, id, root, data = redata)
newdata <- data.frame(pop = levels(echinacea$pop))
for (v in vars)
  newdata[[v]] <- 1
newdata$root <- 1
newdata$ewloc <- 0
newdata$nsloc <- 0
renewdata <- reshape(newdata, varying = list(vars),
                   direction = "long", timevar = "varb", times = as.factor(vars),
                   v.names = "resp")
hdct <- grep("hdct", as.character(renewdata$varb))
hdct <- is.element(seq(along = renewdata$varb), hdct)
renewdata <- data.frame(renewdata, hdct = as.integer(hdct))
beta.hat <- aout4$coef
theta.hat <- predict(aout4, model.type = "cond", parm.type = "canon")
theta.hat <- matrix(theta.hat, nrow = nrow(aout4$x), ncol = ncol(aout4$x))
xstar <- raster(theta.hat, pred, fam, aout4$root)
aout4star <- aster(xstar, aout4$root, pred, fam, aout4$modmat, beta.hat)
beta.star <- aout4star$coef
print(cbind(beta.hat, beta.star))
```

sim

*Simulated Life History Data***Description**

Data on life history traits for four years and five fitness components

Usage

```
data(sim)
```

Format

Loads nine objects. The objects `beta.true`, `mu.true`, `phi.true`, and `theta.true` are the simulation truth parameter values in different parametrizations.

beta.true Regression coefficient vector for model $\text{resp} \sim \text{varb} + 0 + z1 + z2 + I(z1^2) + I(z1*z2) + I(z2^2)$.

mu.true Unconditional mean value parameter vector for same model.

phi.true Unconditional canonical value parameter vector for same model.

theta.true Conditional canonical value parameter vector for same model.

The objects `fam`, `pred`, and `vars` specify the aster model graphical and probabilistic structure.

fam Integer vector giving the families of the variables in the graph.

pred Integer vector giving the predecessors of the variables in the graph.

vars Character vector giving the names of the variables in the graph.

The objects `ladata` and `redata` are the simulated data in two forms "wide" and "long" in the terminology of the `reshape` function.

ladata Data frame with variables `y`, `z1`, `z2` used for Lande-Arnold type estimation of fitness landscape. `y` is the response, fitness, and `z1` and `z1` are predictor variables, phenotypes.

redata Data frame with variables `resp`, `z1`, `z2`, `varb`, `id`, `root` used for aster type estimation of fitness landscape. `resp` is the response, containing all components of fitness, and `z1` and `z1` are predictor variables, phenotypes. `varb` is a factor whose levels are elements of `vars` indicating which elements of `resp` go with which nodes of the aster model graphical structure. The variables `z1` and `z2` have been set equal to zero except when `grep("nseed", varb)` is TRUE. For the rationale see Section 3.2 of TR 669 referenced below.

Source

Geyer, C. J and Shaw, R. G. (2008) Supporting Data Analysis for a talk to be given at Evolution 2008. Technical Report No. 669. School of Statistics, University of Minnesota. <http://www.stat.umn.edu/geyer/aster/>.

References

Geyer, C. J and Shaw, R. G. (2009) Hypothesis Tests and Confidence Intervals Involving Fitness Landscapes fit by Aster Models. Technical Report No. 671. School of Statistics, University of Minnesota. <http://www.stat.umn.edu/geyer/aster/>.

Examples

```
data(sim)
out6 <- aster(resp ~ varb + 0 + z1 + z2 + I(z1^2) + I(z1*z2) + I(z2^2),
  pred, fam, varb, id, root, data = redata)
summary(out6)
lout <- lm(y ~ z1 + z2 + I(z1^2) + I(z1*z2) + I(z2^2), data = ladata)
summary(lout)
```

summary.aster

Summarizing Aster Model Fits

Description

These functions are all [methods](#) for class `aster` or `summary.aster` objects.

Usage

```
## S3 method for class 'aster':
summary(object, info = c("expected", "observed"),
  info.tol = sqrt(.Machine$double.eps), show.graph = FALSE, ...)

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

Arguments

<code>object</code>	an object of class "aster", usually, a result of a call to aster .
<code>info</code>	the type of Fisher information use to compute standard errors.
<code>info.tol</code>	tolerance for eigenvalues of Fisher information. If <code>eval</code> is the vector of eigenvalues of the information matrix, then <code>eval < cond.tol * max(eval)</code> are considered zero. Hence the corresponding eigenvectors are directions of constancy or recession of the log likelihood.
<code>show.graph</code>	if TRUE, show the graphical model.
<code>x</code>	an object of class "summary.aster", usually, a result of a call to <code>summary.aster</code> .
<code>digits</code>	the number of significant digits to use when printing.
<code>signif.stars</code>	logical. If TRUE, "significance stars" are printed for each coefficient.
<code>...</code>	further arguments passed to or from other methods.

Value

`summary.aster` returns an object of class `"summary.aster"` list with the same components as `object`, which is of class `"aster"`.

See Also

[aster](#), [summary](#).

truncated

K-Truncated Distributions

Description

Random generation for the k -truncated Poisson distribution or for the k -truncated negative binomial distribution, where “ k -truncated” means conditioned on being strictly greater than k . If `xpred` is not one, then the random variate is the sum of `xpred` such random variates.

Usage

```
rktp(n, k, mu, xpred = 1)
rktnb(n, size, k, mu, xpred = 1)
rnzp(n, mu, xpred = 1)
```

Arguments

<code>n</code>	number of random values to return. If <code>length(n) > 1</code> , the length is taken to be the number required.
<code>size</code>	the size parameter for the negative binomial distribution.
<code>k</code>	truncation limit.
<code>xpred</code>	number of trials.
<code>mu</code>	vector of positive means.

Details

`rktp` simulates k -truncated Poisson random variates. `rktnb` simulates k -truncated negative binomial random variates. `rnzp` simulates zero-truncated Poisson random variates (maintained only for backward compatibility, it now calls `rktp`).

Value

a vector of random deviates.

See Also

[families](#)

Examples

```
rktp(10, 2, 0.75)  
rktnb(10, 2.222, 2, 0.75)
```

Index

*Topic **datasets**

aphid, 3
chamae, 8
chamae2, 9
echin2, 11
echinacea, 12
sim, 23

*Topic **distribution**

raster, 21
truncated, 25

*Topic **misc**

families, 13
mlogl, 16

*Topic **models**

anova.aster, 1
aster, 4
predict.aster, 17
summary.aster, 24

*Topic **regression**

anova.aster, 1
aster, 4
predict.aster, 17

anova, 2, 7
anova.aster, 1, 7
anova.asterlist (anova.aster), 1
aphid, 3
aster, 2, 4, 13, 15–17, 20, 22, 24, 25
aster.default, 20
aster.formula, 20

beta.true (sim), 23

chamae, 8
chamae2, 9

echin2, 11
echinacea, 12

fam (sim), 23
fam.bernoulli (families), 13

fam.default (families), 13
fam.negative.binomial (families),
13
fam.normal.location (families), 13
fam.poisson (families), 13
fam.truncated.negative.binomial
(families), 13
fam.truncated.poisson (families),
13
famfun (families), 13
families, 5, 13, 17, 22, 25
formula, 5

glm, 5

ladata (sim), 23
lm, 5

methods, 24
mlogl, 13, 15, 16
mu.true (sim), 23

nlm, 5, 7

optim, 5, 7

phi.true (sim), 23
pred (sim), 23
predict, 7
predict.aster, 7, 17
predict.glm, 18
predict.lm, 20
print.summary.aster
(summary.aster), 24

raster, 21
redata (sim), 23
reshape, 5, 7, 20
rktnb (truncated), 25
rktp (truncated), 25
rnzp (truncated), 25

`sim`, [23](#)
`summary`, [7](#), [25](#)
`summary.aster`, [7](#), [24](#)

`terms`, [7](#)
`theta.true(sim)`, [23](#)
`truncated`, [25](#)
`trust`, [5](#), [7](#), [15](#)

`vars(sim)`, [23](#)