Package 'aster'

July 22, 2025

Version 1.3-5

Date 2025-06-02

Title Aster Models

Depends R (>= 4.3.0)

Imports stats, trust

Suggests numDeriv

ByteCompile TRUE

Description Aster models (Geyer, Wagenius, and Shaw, 2007,
<doi:10.1093/biomet/asm030>; Shaw, Geyer, Wagenius, Hangelbroek, and
Etterson, 2008, <doi:10.1086/588063>; Geyer, Ridley, Latta, Etterson,
and Shaw, 2013, <doi:10.1214/13-AOAS653>) are exponential family
regression models for life
history analysis. They are like generalized linear models except that
elements of the response vector can have different families (e. g.,
some Bernoulli, some Poisson, some zero-truncated Poisson, some normal)
and can be dependent, the dependence indicated by a graphical structure.
Discrete time survival analysis, life table analysis,
zero-inflated Poisson regression, and
generalized linear models that are exponential family (e. g., logistic
regression and Poisson regression with log link) are special cases.
Main use is for data in which there is survival over discrete time periods
and there is additional data about what happens conditional on survival
(e. g., number of offspring). Uses the exponential family canonical
parameterization (aster transform of usual parameterization).
There are also random effects versions of these models.

License MIT + file LICENSE

URL https://www.stat.umn.edu/geyer/aster/

NeedsCompilation yes

Author Charles J. Geyer [aut, cre]

Maintainer Charles J. Geyer <geyer@umn.edu>

Repository CRAN

Date/Publication 2025-06-03 06:40:02 UTC

Contents

---

anova.asterOrReaster   *Analysis of Deviance for Reaster Model Fits*

Description

Compute an analysis of deviance table for two or more aster model fits with or without random effects.

Usage

```
## S3 method for class 'asterOrReaster'
anova(object, ...,
    tolerance = .Machine$double.eps ^ 0.75)
anovaAsterOrReasterList(objectlist, tolerance = .Machine$double.eps ^ 0.75)
```

Arguments

`object, ...`	objects of class `"asterOrReaster"`, typically the result of a call to aster or reaster.
`objectlist`	list of objects of class `"asterOrReaster"`.
`tolerance`	tolerance for comparing nesting of model matrices.

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.

For objects of class `"reaster"`, the quantity called deviance is only approximate. See references on help for reaster.

When objects of class `"reaster"` are among those supplied, degrees of freedom for fixed effects and degrees of freedom for variance components are reported separately, because tests for fixed effects are effectively two-tailed and tests for variance components are effectively one-tailed.

In case models being compared differ by one variance component, the reference distribution is half a chi-square with the fixed effect degrees of freedom (difference of number of fixed effects in the two models) and half a chi-square with one more degrees of freedom.

In case models being compared differ by two or more variance components, we do not know how to how to do the test. The reference distribution is a mixture of chi-squares but the mixing weights are difficult to calculate. An error is given in this case.

Value

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

Warning

The comparison between two or more models by anova or anovaAsterOrReasterList will only be valid if they are (1) fitted to the same dataset, (2) models are nested, (3) have the same dependence graph and exponential families. Some of this is currently checked. Some warnings are given.

See Also

aster, reaster, anova.

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 <- grepl("hdct", as.character(redata$varb))
redata <- data.frame(redata, hdct = as.integer(hdct))
level <- gsub("[0-9]", "", as.character(redata$varb))
redata <- data.frame(redata, level = as.factor(level))
aout1 <- aster(resp ~ varb + hdct : (nsloc + ewloc + pop),
    pred, fam, varb, id, root, data = redata)
aout2 <- aster(resp ~ varb + level : (nsloc + ewloc) + hdct : pop,
    pred, fam, varb, id, root, data = redata)
aout3 <- aster(resp ~ varb + level : (nsloc + ewloc + pop),
    pred, fam, varb, id, root, data = redata)
anova(aout1, aout2, aout3)

# now random effects models and models without random effects mixed
## Not run:
### CRAN policy says examples must take < 5 sec.
### This doesn't (on their computers).
data(radish)
pred <- c(0,1,2)
fam <- c(1,3,2)
rout2 <- reaster(resp ~ varb + fit : (Site * Region),
    list(block = ~ 0 + fit : Block, pop = ~ 0 + fit : Pop),
    pred, fam, varb, id, root, data = radish)
rout1 <- reaster(resp ~ varb + fit : (Site * Region),
    list(block = ~ 0 + fit : Block),
    pred, fam, varb, id, root, data = radish)
rout0 <- aster(resp ~ varb + fit : (Site * Region),
    pred, fam, varb, id, root, data = radish)
anova(rout0, rout1, rout2)

## End(Not run)
```

---

aphid   *Life History Data on Uroleucon rudbeckiae*

Description

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

Usage

```
data(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 fecunidity. *Ecology*, **63**, 655-662. doi:10.2307/1936785.

These data are reanalyzed by aster methods in the following.

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2008) Unifying life history analyses for inference of fitness and population growth. *American Naturalist*, **172**, E35-E47. doi:10.1086/588063.

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
: an `nind` by `nnode` matrix, the data for an aster model. The rows are independent and identically modeled random vectors. See details below for further requirements.

  `aster.formula` constructs such an `x` from the response in its formula. Hence data for `aster.formula` must be a vector of length `nind * nnode`.

root
: an object of the same shape as `x`, the root data. For `aster.default` an `nind` by `nnode` matrix, For `aster.formula` an `nind * nnode` vector.

pred
: an integer vector of length `nnode` determining the dependence graph of the aster model. `pred[j]` is the index of the predecessor of the node with index `j` unless the predecessor is a root node, in which case `pred[j] == 0`. See details below for further requirements.

fam
: an integer vector of length `nnode` determining the exponential family structure of the aster model. Each element is an index into the vector of family specifications given by the argument `famlist`.

modmat
: an `nind` by `nnode` by `ncoef` three-dimensional array, the model matrix.

  `aster.formula` constructs such a `modmat` from its formula, the data frame `data`, and the variables in the environment of the formula.

parm
: usually missing. Otherwise a vector of length `ncoef` giving a starting point for the optimization.

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 details below for further explanation. This is what [lm](#), [glm](#) and other functions that do regression call "offset" but we don't change our name for reasons of backward compatibility.

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.

method
: optimization method. If `"trust"` then the [trust](#) function is used. If `"nlm"` then the [nlm](#) function is used. Otherwise the [optim](#) function is used with the specified `method` supplied to it. The value of this argument can be abbreviated.

fscale	an estimate of the size of the log likelihood at the maximum. Defaults to nind.
maxiter	maximum number of iterations. Defaults to '1000'.
nowarn	if TRUE (the default), suppress warnings from the optimization routine.
newton	if TRUE (the default), do one Newton iteration on the result produced by the optimization routine, except when method == "trust" when no such Newton iteration is done, regardless of the value of newton, because [trust](#) always terminates with a Newton iteration when it converges.
optout	if TRUE, save the entire result of the optimization routine ([trust](#), [nlm](#), or [optim](#), as the case may be).
coef.names	names of the regression coefficients. If missing, dimnames(modmat)[[3]] is used. In aster.formula these are produced automatically by the R formula machinery.
...	other arguments passed to the optimization method.
formula	a symbolic description of the model to be fit. See [lm](#), [glm](#), and [formula](#) for discussions of the R formula mini-language.
varvar	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 nnode. Must be of the form rep(vars, each = nind) where vars is a vector of variable names. Usually found in the data frame data when this is produced by the [reshape](#) function.
idvar	a variable of the same length as the response in the formula that indexes individuals. The number of individuals is nind. Must be of the form rep(inds, times = nnode) where inds is a vector of labels for individuals. Usually found in the data frame data 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. This is shown in the examples section below and in the package vignette titled "Aster Package Tutorial".

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:

coefficients   a named vector of coefficients.

rank	the numeric rank of the fitted generalized linear model part of the aster model (i.e., the rank of `modmat`).
deviance	up to a constant, minus twice the maximized log-likelihood. (Note the minus. This is somewhat counterintuitive, but cannot be changed for reasons of backward compatibility.)
iter	the number of iterations used by the optimization method.
converged	logical. Was the optimization algorithm judged to have converged?
code	integer. The convergence code returned by the optimization method.
gradient	The gradient vector of minus the log likelihood at the fitted `coefficients` vector.
hessian	The Hessian matrix of minus the log likelihood (i.e., the observed Fisher information) at the fitted `coefficients` vector. This is also the expected Fisher information when `type == "unconditional"`.
fisher	Expected Fisher information at the fitted `coefficients` vector.
optout	The object returned by the optimization routine (`trust`, `nlm`, or `optim`). Only returned when the argument `optout` is `TRUE`.

Calls to `aster.formula` return a list also containing:

call	the matched call.
formula	the formula supplied.
terms	the `terms` object used.
data	the `data` argument.

NA Values

It was almost always wrong for aster model data to have `NA` values. Although theoretically possible for the R formula mini-language to do the right thing for an aster model with `NA` values in the data, usually it does some wrong thing. Thus, since version 0.8-20, this function and the `reaster` function give errors when used with data having `NA` values. Users must remove all `NA` values (or replace them with what they should be, perhaps zero values) "by hand".

Directions of Recession

Even if the result of this function has component `converges` equal to `TRUE`, the result will be nonsense if there are one or more directions of recession. These are not detected by this function, but rather by the `summary` function applied to the result of this function, for which see `summary.aster`.

References

Geyer, C. J., Wagenius, S., and Shaw, R. G. (2007) Aster Models for Life History Analysis. *Biometrika*, **94**, 415–426. doi:10.1093/biomet/asm030.

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2008) Unifying Life History Analysis for Inference of Fitness and population growth. *American Naturalist*, **172**, E35–E47. doi: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 <- grepl("hdct", as.character(redata$varb))
redata <- data.frame(redata, hdct = as.integer(hdct))
level <- gsub("[0-9]", "", as.character(redata$varb))
redata <- data.frame(redata, level = as.factor(level))
aout <- aster(resp ~ varb + level : (nsloc + ewloc) + hdct : pop,
    pred, fam, varb, id, root, data = redata)
summary(aout, show.graph = TRUE)
```

---

astertransform   *Transform between Aster Model Parameterizations*

Description

Transform between different parameterizations of the aster model. In effect, this function is called inside predict.aster. Users generally do not need to call it directly.

Usage

```
astertransform(arg, obj, from = c("unconditional", "conditional"),
    to.cond = c("unconditional", "conditional"),
    to.mean = c("mean.value", "canonical"))
```

Arguments

arg   canonical parameter vector of length nrow(obj$data), either unconditional (φ)
             or conditional (θ) depending on the value of argument from.

obj   aster model object, the result of a call to aster.

from   the type of canonical parameter which argument arg is.

to.cond   the type of parameter we want.

to.mean   the type of parameter we want.

Value

a vector of the same length as arg, the transformed parameter vector.

---

chamae   *Life History Data on Chamaecrista fasciculata*

Description

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

Usage

```
data(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 https://sites.google.com/d.umn.edu/dr-julie-r-etterson/home

References

These data are a subset of data 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. doi:10.1111/j.00143820.2004.tb01726.x.

Etterson, J. R., and Shaw, R. G. (2001). Constraint to adaptive evolution in response to global warming. *Science*, **294**, 151-154. doi:10.1126/science.1063656.

These data are reanalyzed by aster methods in the following.

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2008) Unifying life history analyses for inference of fitness and population growth. *American Naturalist*, **172**, E35-E47. doi:10.1086/588063.

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

```
data(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 https://sites.google.com/d.umn.edu/dr-julie-r-etterson/home

References

These data are a subset of data 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. doi:10.1111/j.00143820.2004.tb01726.x.

Etterson, J. R., and Shaw, R. G. (2001). Constraint to adaptive evolution in response to global warming. *Science*, **294**, 151-154. doi:10.1126/science.1063656.

These data are reanalyzed by aster methods in the following.

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2008) Unifying life history analyses for inference of fitness and population growth. *American Naturalist*, **172**, E35-E47. doi:10.1086/588063.

Examples

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

---

chamae3 *Life History Data on Chamaecrista fasciculata*

Description

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

Usage

```
data(chamae3)
```

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.

fit Zero-or-one-valued. Indicates "fitness" nodes of the graph.

SIRE Categorical. Sire.

DAM Categorical. Dam.

SITE Categorical. Experiment site.

POP Categorical. Population of sire and dam.

ROW Numerical. Row. Position in site.

BLK Categorical. Block within site.

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 https://sites.google.com/d.umn.edu/dr-julie-r-etterson/home

References

These data are a subset of data previously analyzed by non-aster methods in the following.

Etterson, J. R. (2004). Evolutionary potential of *Chamaecrista fasciculata* in relation to climate change. II. Genetic architecture of three populations reciprocally planted along an environmental gradient in the great plains. *Evolution*, **58**, 1459–1471. doi:10.1111/j.00143820.2004.tb01727.x.

These data were reanalyzed by aster methods in the following.

Geyer, C. J., Ridley, C. E., Latta, R. G., Etterson, J. R., and Shaw, R. G. (2013) Local Adaptation and Genetic Effects on Fitness: Calculations for Exponential Family Models with Random Effects. *Annals of Applied Statistics*, **7**, 1778–1795. doi:10.1214/13AOAS653.

Examples

```
data(chamae3)
### wide version
## Not run:
### CRAN policy says examples must take < 5 sec.  This doesn't.
foo <- chamae3
### delete fit because it makes no sense in wide version
foo$fit <- NULL
chamae3w <- reshape(foo, direction = "wide", timevar = "varb",
    v.names = "resp", varying = list(levels(chamae3$varb)))

## End(Not run)
```

---

echin2   *Life History Data on Echinacea angustifolia*

Description

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

Usage

```
data(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, https://www.chicagobotanic.org/research/staff/wagenius

References

These data are analyzed in the following.

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2008) Unifying life history analyses for inference of fitness and population growth. *American Naturalist*, **172**, E35-E47. doi:10.1086/588063.

Examples

```
data(echin2)
```

---

echinacea	*Life History Data on Echinacea angustifolia*

Description

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

Usage

```
data(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, https://www.chicagobotanic.org/research/staff/wagenius

References

These data are analyzed in the following.

Geyer, C. J., Wagenius, S., and Shaw, R. G. (2007) Aster Models for Life History Analysis. *Biometrika*, **94**, 415–426. doi:10.1093/biomet/asm030.

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

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

---

foobar   *Toy Life History Data having Directions of Recession*

Description

Toy life history data created to exhibit the phenomenon of directions of recession. It was analyzed in a special topics course on aster models Fall 2018.

Usage

```
data(foobar)
```

Format

`data(foobar)` loads four R objects.

fam a vector of family indices.

pred a vector of predecessor indices.

vars a vector of variable names associated with the nodes of the aster graph.

redata a data frame having data on 300 individuals, each of which has `length(vars) == 4` components of fitness, so the aster graph for one individual has four nodes. This data frame is already in long format; no need to `reshape`. The variables in this data frame are:

     resp the 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. Individual ID numbers.

     trt Categorical. Treatment.

     blk Categorical. Block.

     fit Bernoulli (zero-or-one-valued). Indicator variable of the fitness nodes of the graph; in these data there is just one node for fitness.

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 the corresponding component of the response vector was. The variable `id` indicates which individual the corresponding component of the response vector was.

Source

Charles J. Geyer https://www.stat.umn.edu/geyer/8931aster/foobar.rda

References

These data are analyzed in deck 9 of the slides for a special topics course on aster models taught
fall semester 2018 (https://www.stat.umn.edu/geyer/8931aster/slides/s9.pdf).

Examples

```
data(foobar)
library(aster)
aout <- aster(resp ~ varb + fit : (trt * blk), pred,
    fam, varb, id, root, data = redata)
foo <- try(summary(aout))
# gives an error about directions of recession
# get directions of recession
dor <- attr(foo, "condition")$dor
dor
# found one apparent direction of recession
# from regular pattern
# it looks like a true direction of recession
dor <- dor / max(abs(dor))
dor
# but what does it do?  For that need to map to saturated model
# parameter space
modmat <- aout$modmat
dim(modmat)
# oof!  modmat is three-dimensional.  Need an actual matrix here.
modmat <- matrix(as.vector(modmat), ncol = length(dor))
dor.phi <- drop(modmat %*% dor)
names(dor.phi) <- with(redata, paste(id, as.character(varb), sep = "."))
dor.phi[dor.phi != 0]
fam.default()[fam[vars == "seeds"]]
# since the support of the Poisson distribution is bounded above,
# actually this must be minus the DOR (if it is a DOR at all).
# check that all components of response vector for which dor.phi == 1 are zero
# (lower bound of Poisson range)
all(redata$resp[dor.phi == 1] == 0)
# so minus dor.phi is a true direction of recession in the saturated model
# canonical parameter space, and minus dor is a true direction of recession
# in the submodel canonical parameter space
#
# try to get more info
trt.blk <- with(redata,
    paste(as.character(trt), as.character(blk), sep = "."))
unique(trt.blk[dor.phi == 1])
# the reason for the direction of recession is that every individual getting
# treatment a in block A had zero seeds.
#
# the reason the submodel DOR, R object dor, was so hard to interpret was
# because fit:trta:blkA is not in the model.  So let's force it in
redata <- transform(redata, trt = relevel(trt, ref = "b"),
    blk = relevel(blk, ref = "B"))
# Note: following code is copied exactly from above.  Only difference
# is releveling in the immediately preceding statement
```

```
aout <- aster(resp ~ varb + fit : (trt * blk), pred,
    fam, varb, id, root, data = redata)
foo <- try(summary(aout))
dor <- attr(foo, "condition")$dor
dor <- dor / max(abs(dor))
dor
# now it is obvious from looking at this dor that all individuals in trt a
# and blk A are at the lower end (zero) of the Poisson range.
# maybe the other dor we had previously would be "obvious" to someone
# sufficiently skilled in understanding the meaning of regression coefficients
# but not "obvious" to your humble author
#
# as for what to do about this, see the course slides cited in the reference
# section.  There is no single Right Thing to do.
```

---

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 R function [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 length(parm) as ncoef.
pred	integer vector determining the graph. pred[j] is the index of the predecessor of the node with index j unless the predecessor is a root node, in which case pred[j] == 0. We also refer to length(pred) as nnode. This argument is required to satisfy pred[j] < j for all j.
fam	an integer vector of length nnode determining the exponential family structure of the aster model. Each element is an index into the vector of family specifications given by the argument famlist.
x	the response. If a matrix, rows are individuals, and columns are variables (nodes of graphical model). So ncol(x) == nnode and we also refer to nrow(x) as nind. If not a matrix, then x must be as if it were such a matrix and then dimension information removed by x = as.numeric(x).
root	A matrix or vector like x. Data root[i, j] is the data for a root node that is the predecessor of the response x[i, j] and is ignored when pred[j] > 0.

modmat	a three-dimensional array, `nind` by `nnode` by `ncoef`, the model matrix. Or a matrix, `nind * nnode` by `ncoef` (when x and `root` are one-dimensional of length `nind * nnode`).
deriv	derivatives 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).

---

newpickle   *Penalized Quasi-Likelihood for Aster Models*

Description

Evaluates the objective function for approximate maximum likelihood for an aster model with random effects. Uses Laplace approximation to integrate out the random effects analytically. The "quasi" in the title is a misnomer in the context of aster models but the acronym PQL for this procedure is well-established in the generalized linear mixed models literature.

Usage

```
newpickle(alphaceesigma, fixed, random, obj, y, origin, zwz, deriv = 0)
```

Arguments

alphaceesigma	the parameter value where the function is evaluated, a numeric vector, see details.
fixed	the model matrix for fixed effects. The number of rows is `nrow(obj$data)`. The number of columns is the number of fixed effects.
random	the model matrix or matrices for random effects. The number of rows is `nrow(obj$data)`. The number of columns is the number of random effects in a group. Either a matrix or a list each element of which is a matrix.

obj	aster model object, the result of a call to [aster](#).
y	response vector. May be omitted, in which case obj$x is used. If supplied, must be a matrix of the same dimensions as obj$x.
origin	origin of aster model. May be omitted, in which case default origin (see [aster](#)) is used. If supplied, must be a matrix of the same dimensions obj$x.
zwz	A possible value of $Z^T W Z$, where Z is the model matrix for all random effects and W is the variance matrix of the response. May be missing, in which case it is calculated from alphaceesigma. See details.
deriv	Number of derivatives wanted, either zero or one. Must be zero if zwz is missing.

Details

Define

$$p(\alpha, c, \sigma) = m(a + M\alpha + ZAc) + c^T c / 2 + \log \det[AZ^T W(a + M\alpha + ZAc)ZA + I]$$

where m is minus the log likelihood function of a saturated aster model, where W is the Hessian matrix of m, where a is a known vector (the *offset vector* in the terminology of [glm](#) but the *origin* in the terminology of [aster](#)), where M is a known matrix, the model matrix for fixed effects (the argument fixed of this function), Z is a known matrix, the model matrix for random effects (either the argument random of this functions if it is a matrix or Reduce(cbind, random) if random is a list of matrices), where A is a diagonal matrix whose diagonal is the vector rep(sigma, times = nrand) where nrand is sapply(random, ncol) when random is a list of matrices and ncol(random) when random is a matrix, and where I is the identity matrix. This function evaluates $p(\alpha, c, \sigma)$ when zwz is missing. Otherwise it evaluates the same thing except that

$$Z^T W(a + M\alpha + ZAc)Z$$

is replaced by zwz. Note that A is a function of σ although the notation does not explicitly indicate this.

Value

a list with components value and gradient, the latter missing if deriv == 0.

Note

Not intended for use by naive users. Use [reaster](#). Actually no longer used by other functions in this package.

Examples

```
data(radish)

pred <- c(0,1,2)
fam <- c(1,3,2)

### need object of type aster to supply to penmlogl and pickle

aout <- aster(resp ~ varb + fit : (Site * Region + Block + Pop),
```

```
        pred, fam, varb, id, root, data = radish)

### model matrices for fixed and random effects

modmat.fix <- model.matrix(resp ~ varb + fit : (Site * Region),
    data = radish)
modmat.blk <- model.matrix(resp ~ 0 + fit:Block, data = radish)
modmat.pop <- model.matrix(resp ~ 0 + fit:Pop, data = radish)

rownames(modmat.fix) <- NULL
rownames(modmat.blk) <- NULL
rownames(modmat.pop) <- NULL

idrop <- match(aout$dropped, colnames(modmat.fix))
idrop <- idrop[! is.na(idrop)]
modmat.fix <- modmat.fix[ , - idrop]

nfix <- ncol(modmat.fix)
nblk <- ncol(modmat.blk)
npop <- ncol(modmat.pop)

alpha.start <- aout$coefficients[match(colnames(modmat.fix),
    names(aout$coefficients))]
cee.start <- rep(0, nblk + npop)
sigma.start <- rep(1, 2)
alphaceesigma.start <- c(alpha.start, cee.start, sigma.start)

foo <- newpickle(alphaceesigma.start, fixed = modmat.fix,
    random = list(modmat.blk, modmat.pop), obj = aout)
```

---

oats   *Life History Data on Avena barbata*

Description

Data on life history traits for the invasive California wild oat *Avena barbata*

Usage

```
data(oats)
```

Format

A data frame with records for 821 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.

Plant.id Categorical. Another indicator of individual plants.

Env Categorical. Environment in which plant was grown, a combination of experimental site and year.

Gen Categorical. Ecotype of plant: mesic (M) or xeric (X).

Fam Categorical. Accession, nested within ecotype.

Site Categorical. Experiment site. Two sites in these data.

Year Categorical. Year in which data were collected. Four years in these data.

fit Indicator (zero or one). Shorthand for `as.numeric(oats$varb == "Spike")`. So-called because the components of `outcome` indicated are the best surrogate of Darwinian fitness in these data.

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

Surv Indicator (zero or one). Bernoulli, One if individual survived to produce flowers.

Spike Integer. Zero-truncated Poisson, number of spikelets (compound floral structures) observed.

Graphical model is

$$1 \longrightarrow \text{Surv} \longrightarrow \text{Spike}$$

Source

Robert Latta https://www.dal.ca/faculty/science/biology/faculty-staff/our-faculty/robert-latta.html

References

These data are a subset of data previously analyzed using non-aster methods in the following.

Latta, R. G. (2009). Testing for local adaptation in *Avena barbata*, a classic example of ecotypic divergence. *Molecular Ecology*, **18**, 3781–3791. doi:10.1111/j.1365294X.2009.04302.x.

These data are reanalyzed by aster methods in the following.

Geyer, C. J., Ridley, C. E., Latta, R. G., Etterson, J. R., and Shaw, R. G. (2013) Local Adaptation and Genetic Effects on Fitness: Calculations for Exponential Family Models with Random Effects. *Annals of Applied Statistics*, **7**, 1778–1795. doi:10.1214/13AOAS653.

Examples

```
data(oats)
```

---

objfun   *Function Factory that Makes Objective Function for Aster Models with Random Effects*

Description

A function factory that returns an R function that evaluates the objective function for fitting aster models with random effects

Usage

```
objfun.factory(fixed, random, response, pred, fam, root, zwz,
    famlist = fam.default(), offset, standard.deviation = FALSE,
    deriv = 0)
```

Arguments

fixed	the model matrix for fixed effects.
random	either a model matrix or list of model matrices. Each model matrix operates on the random effects for one variance component. See Details section for more.
response	a vector specifying the response. Length is the row dimension of all model matrices.
pred	an integer vector of length nnode determining the dependence graph of the aster model. pred[j] is the index of the predecessor of the node with index j unless the predecessor is a root node, in which case pred[j] == 0. See details section of [aster] for further requirements.
fam	an integer vector of length nnode determining the exponential family structure of the aster model. Each element is an index into the vector of family specifications given by the argument famlist.
root	a vector whose length is the row dimension of all model matrices. For nodes whose predecessors are root nodes specifies the value of the constant at that root node. Typically the vector having all components equal to one. Always positive integer valued. If components greater than one, then graph for that "individual" in scare quotes is actually for data for multiple individuals.
zwz	a positive definite symmetric matrix (neither symmetry nor positive definiteness is checked, but incorrect answers with no error or warning will result if condition is not met) whose dimensions are the number of random effects. Should be $Z^T W(\varphi) Z$ where Z is the model matrix for random effects and $W(\cdot)$ is minus the second derivative of the saturated model aster log likelihood with respect to its unconditional canonical parameter. But it does not have to be. Use R function [makezwz] to make this.
famlist	a list of family specifications (see [families]).
offset	a vector whose length is the row dimension of all model matrices. Distinguished point in parameter space. May be missing, in which case an unspecified default is provided. See details of [aster] for further explanation, where it is called "origin" rather than "offset".

standard.deviation

> scalar logical. If `FALSE`, use the parameterization alpha, b, nu, fixed effects, random effects, and variance components, respectively. If `TRUE`, use the parameterization alpha, c, `sigma`. See Details for more about these parameterizations.

deriv

> 0, 1, or 2. Number of derivatives. Zero means only the function value, one means gradient, two means Hessian.

Details

See the help page for the function aster for specification of aster models. This function only fits unconditional aster models (those with default values of the aster function arguments `type` and `origin.type`.

The only difference between this function and functions `aster` and `mlogl` is that some effects (also called coefficients, also called canonical affine submodel canonical parameters) are treated as random. The unconditional canonical parameter vector of the (saturated) aster model is treated as an affine function of fixed and random effects

$$\varphi = a + M\alpha + \sum_{i=1}^{k} Z_i b_i$$

where M and the Z_i are model matrices specified by the arguments `fixed` and `random`, where α is a vector of fixed effects and each b_i is a vector of random effects that are assumed to be (marginally) normally distributed with mean vector zero and variance matrix ν_i times the identity matrix. The random effects vectors b_i are assumed independent.

In what follows, Z is the single matrix produced by cbind'ing the Z_i, and b is the single vector produced by c'ing the b_i, and ν is the single vector produced by c'ing the ν_i, and D is the variance-covariance matrix of b, which is diagonal with all diagonal components equal to some component of ν.

We can write

$$D = \sum_k \nu_k E_k$$

where the nus are unknown parameters to estimate and the Es are known diagonal matrices whose components are zero-or-one-valued having the property that they sum to the identity matrix and multiply to the zero matrix (so each component of b is exactly one component of ν).

The R function returned by this function evaluates

$$p_W(\alpha, b, \nu) = -l(\varphi) + \tfrac{1}{2} b^T D^{-1} b + \tfrac{1}{2} \log \det \left(Z^T W Z D + \mathrm{Id} \right)$$

where l is the log likelihood for the saturated aster model specified by arguments `pred`, `fam`, and `famlist`, where φ is given by the displayed equation above, and where $Z^T W Z$ is argument zwz. This is equation (7) in Geyer, et al. (2013).

The displayed equation above only defines p_W when all of the variance components (components of ν and diagonal components of D) are strictly positive because otherwise D is not invertible. In order to be lower semicontinuous, the R function returned by this function returns `Inf` whenever any component of ν is nonpositive, except it returns zero if $b_i = 0$ whenever the corresponding diagonal element of D is zero.

When argument `standard.deviation` is `TRUE`, the arguments to the objective function are replaced by the change-of-variables

$$\nu = \sigma^2$$

and

$$b = Ac$$

where the first equation works componentwise (as in R vector arithmetic), components of the vector σ are allowed to be negative, and $D = A^2$ so

$$A = \sum_k \sigma_k E_k$$

When the $(\alpha.c, \sigma)$ variables are used, p_W is continuous and infinitely differentiable for all values of these variables.

When the $(\alpha.b, \nu)$ variables are used, p_W is continuous and infinitely differentiable for all values of α and b but only strictly positive values ν. When some components of ν are zero, then subderivatives exist, but not derivatives (Geyer, et al., 2013).

Value

returns an R function having one argument which is a numeric vector of all the parameters and random effects in the order (α, b, ν) for one set of variables or (α, c, σ) for other set.

The R function returned by this function evaluates the function p_W defined in the Details section and returns a list with components named `value`, `gradient`, and `hessian` in case the `deriv` argument is 2. The returned list has only the first two of these in case the `deriv` argument is 1. The returned list has only the first one of these in case the `deriv` argument is 0.

In case the (α, b, ν) parameterization is being used and some components of ν are zero, the derivatives returned are for the terms of the objective function that are differentiable.

Warning about Negative Binomial

The negative binomial and truncated negative binomial families are fundamentally incompatible with random effects. The reason is that the canonical parameter space for a one-parameter negative binomial or truncated negative binomial is the negative half line. Thus the conditional canonical parameter θ for such a node must be negative valued. The aster transform is so complicated that it is unclear what the corresponding constraint on the unconditional canonical parameter φ is, but there is a constraint: its parameter space is not the whole real line. A normal random effect, in contrast, does have support the whole real line. It wants to make parameters that are constrained to have any real number. The code only warns about this situation, because if the random effects do not influence any negative binomial or truncated negative binomial nodes of the graph, then there would be no problem.

Warning about Individual Random Effects

The Breslow-Clayton approximation assumes the complete data log likelihood is approximately quadratic considered as a function of random effects only. This will be the case by the law of large numbers if the number of individuals is much larger than the number of random effects. Thus Geyer, et al. (2013) warn against trying to put a random effect for each individual in the model. If you do that, the code will try to fit the model, but it will take forever and no theory says the results will make any sense.

References

Breslow, N. E., and Clayton, D. G. (1993). Approximate Inference in Generalized Linear Mixed Models. *Journal of the American Statistical Association*, **88**, 9–25. doi:10.1080/01621459.1993.10594284.

Geyer, C. J., Ridley, C. E., Latta, R. G., Etterson, J. R., and Shaw, R. G. (2013) Local Adaptation and Genetic Effects on Fitness: Calculations for Exponential Family Models with Random Effects. *Annals of Applied Statistics*, **7**, 1778–1795. doi:10.1214/13AOAS653.

Examples

```
library(aster)
data(radish)
pred <- c(0,1,2)
fam <- c(1,3,2)
rout <- reaster(resp ~ varb + fit : (Site * Region),
    list(block = ~ 0 + fit : Block, pop = ~ 0 + fit : Pop),
    pred, fam, varb, id, root, data = radish)
objfun <- with(rout, objfun.factory(fixed, random, response,
    obj$pred, obj$fam, as.vector(obj$root), zwz))
theta.hat <- with(rout, c(alpha, b, nu))
all.equal(objfun(theta.hat)$value, rout$deviance / 2)
```

---

penmlogl	*Penalized Minus Log Likelihood for Aster Models*

Description

Penalized minus log likelihood for an aster model, and its first and second derivative. The penalization allows for (approximate) random effects. These functions are called inside pickle, pickle1, pickle2, pickle3, and reaster.

Usage

```
penmlogl(parm, sigma, fixed, random, obj, y, origin, deriv = 2)
penmlogl2(parm, alpha, sigma, fixed, random, obj, y, origin)
```

Arguments

parm	for penmlogl, parameter value (vector of regression coefficients and rescaled random effects) at which we evaluate the penalized log likelihood. For penmlogl2 the vector of rescaled random effects only (see next item).
alpha	the vector of fixed effects. For penmlogl2, the concatenation c(alpha, parm) is the same as parm that is supplied to pemnmlogl.
sigma	vector of square roots of variance components, one component for each group of random effects.
fixed	the model matrix for fixed effects. The number of rows is nrow(obj$data). The number of columns is the number of fixed effects.

random	the model matrix or matrices for random effects. Each has the same number of rows as fixed. The number of columns is the number of random effects in a group. Either a matrix or a list of matrices.
obj	aster model object, the result of a call to aster.
y	response vector. May be omitted, in which case obj$x is used. If supplied, must be a matrix of the same dimensions as obj$x.
origin	origin of aster model. May be omitted, in which case default origin (see aster) is used. If supplied, must be a matrix of the same dimensions obj$x.
deriv	number of derivatives wanted. Allowed values are 0, 1, or 2.

Details

Consider an aster model with random effects and canonical parameter vector of the form

$$M\alpha + Z_1 b_1 + \cdots + Z_k b_k$$

where M and each Z_j are known matrices having the same row dimension, where α is a vector of unknown parameters (the fixed effects) and each b_j is a vector of random effects that are supposed to be (marginally) independent and identically distributed mean-zero normal with variance sigma[j]^2.

These functions evaluate minus the "penalized log likelihood" for this model, which considers the random effects as parameters but adds a penalization term

$$b_1^2/(2\sigma_1^2) + \cdots + b_k^2/(2\sigma_k^2)$$

to minus the log likelihood.

To properly deal with random effects that are zero, random effects are rescaled by their standard deviation. The rescaled random effects are $c_i = b_i/\sigma_i$. If $\sigma_i = 0$, then the corresponding rescaled random effects c_i are also zero.

Value

a list containing some of the following components:

value	minus the penalized log likelihood.
gradient	minus the first derivative vector of the penalized log likelihood.
hessian	minus the second derivative matrix of the penalized log likelihood.
argument	the value of the parm argument for this function.
scale	the vector by which parm must be scaled to obtain the true random effects.
mlogl.gradient	gradient for evaluation of log likelihood; gradient is this plus gradient of penalty.
mlogl.hessian	hessian for evaluation of log likelihood; hessian is this plus hessian of penalty.

Note

Not intended for use by naive users. Use reaster, which calls them.

See Also

For an example using this function see the example for pickle.

Description

Evaluates an approximation to minus the log likelihood for an aster model with random effects. Uses Laplace approximation to integrate out the random effects analytically. The "quasi" in the title is a misnomer in the context of aster models but the acronym PQL for this procedure is well-established in the generalized linear mixed models literature.

Usage

```
pickle(sigma, parm, fixed, random, obj, y, origin, cache, ...)
makezwz(sigma, parm, fixed, random, obj, y, origin)
pickle1(sigma, parm, fixed, random, obj, y, origin, cache, zwz,
    deriv = 0, ...)
pickle2(alphasigma, parm, fixed, random, obj, y, origin, cache, zwz,
    deriv = 0, ...)
pickle3(alphaceesigma, fixed, random, obj, y, origin, zwz, deriv = 0)
```

Arguments

sigma	vector of square roots of variance components, one component for each group of random effects. Negative values are allowed; the vector of variance components is `sigma^2`.
parm	starting value for inner optimization. Ignored if `cache$parm` exists, in which case the latter is used. For `pickle` and `pickle1` and `makezwz`, length is number of effects (fixed and random). For `pickle2`, length is number of random effects. For all, random effects are rescaled, divided by the corresponding component of `sigma` if that is nonzero and equal to zero otherwise.
alphasigma	the concatenation of the vector of fixed effects and the vector of square roots of variance components.
alphaceesigma	the concatenation of the vector of fixed effects, the vector of rescaled random effects, and the vector of square roots of variance components.
fixed	the model matrix for fixed effects. The number of rows is `nrow(obj$data)`. The number of columns is the number of fixed effects.
random	the model matrix or matrices for random effects. The number of rows is `nrow(obj$data)`. The number of columns is the number of random effects in a group. Either a matrix or a list each element of which is a matrix.
obj	aster model object, the result of a call to [aster](#).
y	response vector. May be omitted, in which case `obj$x` is used. If supplied, must be a matrix of the same dimensions as `obj$x`.
origin	origin of aster model. May be omitted, in which case default origin (see [aster](#)) is used. If supplied, must be a matrix of the same dimensions `obj$x`.

cache	If not missing, an environment in which to cache the value of `parm` found during previous evaluations. If supplied `parm` is taken from `cache`.
zwz	A possible value of $Z^T W Z$, where Z is the model matrix for all random effects and W is the variance matrix of the response.
deriv	Number of derivatives wanted. For `pickle1` or `pickle2`, either zero or one. For `pickle3`, zero, one or two.
...	additional arguments passed to [trust](#), which is used to maximize the penalized log likelihood.

Details

Define
$$p(\alpha, c, \sigma) = m(a + M\alpha + ZAc) + c^T c/2 + \log \det[AZ^T \widehat{W} ZA + I]/2$$

where m is minus the log likelihood function of a saturated aster model, a is a known vector (the *offset vector* in the terminology of [glm](#) but the *origin* in the terminology of [aster](#)), M is a known matrix, the model matrix for fixed effects (the argument `fixed` of these functions), Z is a known matrix, the model matrix for random effects (either the argument `random` of these functions if it is a matrix or Reduce(cbind, random) if `random` is a list of matrices), A is a diagonal matrix whose diagonal is the vector rep(sigma, times = nrand) where nrand is sapply(random, ncol) when `random` is a list of matrices and ncol(random) when `random` is a matrix, $\widehat{W}$ is any symmetric positive semidefinite matrix (more on this below), and I is the identity matrix. Note that A is a function of σ although the notation does not explicitly indicate this.

Let c^* denote the minimizer of $p(\alpha, c, \sigma)$ considered as a function of c for fixed α and σ, and let $\tilde{\alpha}$ and $\tilde{c}$ denote the (joint) minimizers of $p(\alpha, c, \sigma)$ considered as a function of α and c for fixed σ. Note that c^* is a function of α and σ although the notation does not explicitly indicate this. Note that $\tilde{\alpha}$ and $\tilde{c}$ are functions of σ (only) although the notation does not explicitly indicate this. Now define
$$q(\alpha, \sigma) = p(\alpha, c^*, \sigma)$$

and
$$r(\sigma) = p(\tilde{\alpha}, \tilde{c}, \sigma)$$

Then `pickle1` evaluates $r(\sigma)$, `pickle2` evaluates $q(\alpha, \sigma)$, and `pickle3` evaluates $p(\alpha, c, \sigma)$, where $Z^T \widehat{W} Z$ in the formulas above is specified by the argument `zwz` of these functions. All of these functions supply derivative (gradient) vectors if `deriv = 1` is specified, and `pickle3` supplies the second derivative (Hessian) matrix if `deriv = 2` is specified.

Let W denote the second derivative function of m, that is, $W(\varphi)$ is the second derivative matrix of the function m evaluated at the point φ. The idea is that $\widehat{W}$ should be approximately the value of $W(a + M\hat{\alpha} + Z\widehat{A}\hat{c})$, where $\hat{\alpha}$, $\hat{c}$, and $\hat{\sigma}$ are the (joint) minimizers of $p(\alpha, c, \sigma)$ and $\widehat{A} = A(\hat{\sigma})$. In aid of this, the function `makezwz` evaluates $Z^T W(a + M\alpha + ZAc)Z$ for any α, c, and σ.

`pickle` evaluates the function
$$s(\sigma) = m(a + M\tilde{\alpha} + ZA\tilde{c}) + \tilde{c}^T \tilde{c}/2 + \log \det[AZ^T W(a + M\tilde{\alpha} + ZA\tilde{c})ZA + I]$$

no derivatives can be computed because no derivatives of the function W are computed for aster models.

The general idea is the one uses `pickle` with a no-derivative optimizer, such as the "Nelder-Mead" method of the `optim` function to get a crude estimate of $\hat{\sigma}$. Then one uses [trust](#) with objective

function [penmlogl](#) to estimate the corresponding $\hat{\alpha}$ and $\hat{c}$ (example below). Then one use makezwz to produce the corresponding zwz (example below). These estimates can be improved using [trust](#) with objective function pickle3 using this zwz (example below), and this step may be iterated until convergence. Finally, [optim](#) is used with objective function pickle2 to estimate the Hessian matrix of $q(\alpha, \sigma)$, which is approximate observed information because $q(\alpha, \sigma)$ is approximate minus log likelihood.

Value

For pickle, a scalar, minus the (PQL approximation of) the log likelihood. For pickle1 and pickle2, a list having components value and gradient (present only when deriv = 1). For pickle3, a list having components value, gradient (present only when deriv >= 1), and hessian (present only when deriv = 2).

Note

Not intended for use by naive users. Use [reaster](#), which calls them.

Examples

```
data(radish)
library(trust)

pred <- c(0,1,2)
fam <- c(1,3,2)

### need object of type aster to supply to penmlogl and pickle

aout <- aster(resp ~ varb + fit : (Site * Region + Block + Pop),
    pred, fam, varb, id, root, data = radish)

### model matrices for fixed and random effects

modmat.fix <- model.matrix(resp ~ varb + fit : (Site * Region),
    data = radish)
modmat.blk <- model.matrix(resp ~ 0 + fit:Block, data = radish)
modmat.pop <- model.matrix(resp ~ 0 + fit:Pop, data = radish)

rownames(modmat.fix) <- NULL
rownames(modmat.blk) <- NULL
rownames(modmat.pop) <- NULL

idrop <- match(aout$dropped, colnames(modmat.fix))
idrop <- idrop[! is.na(idrop)]
modmat.fix <- modmat.fix[ , - idrop]

nfix <- ncol(modmat.fix)
nblk <- ncol(modmat.blk)
npop <- ncol(modmat.pop)

### try penmlogl
```

```
sigma.start <- c(1, 1)

alpha.start <- aout$coefficients[match(colnames(modmat.fix),
    names(aout$coefficients))]
parm.start <- c(alpha.start, rep(0, nblk + npop))

tout <- trust(objfun = penmlogl, parm.start, rinit = 1, rmax = 10,
    sigma = sigma.start, fixed = modmat.fix,
    random = list(modmat.blk, modmat.pop), obj = aout)
tout$converged

### crude estimate of variance components

eff.blk <- tout$argument[seq(nfix + 1, nfix + nblk)]
eff.pop <- tout$argument[seq(nfix + nblk + 1, nfix + nblk + npop)]
sigma.crude <- sqrt(c(var(eff.blk), var(eff.pop)))

### try optim and pickle

cache <- new.env(parent = emptyenv())
oout <- optim(sigma.crude, pickle, parm = tout$argument,
    fixed = modmat.fix, random = list(modmat.blk, modmat.pop),
    obj = aout, cache = cache)
oout$convergence == 0
### estimated variance components
oout$par^2

### get estimates of fixed and random effects

tout <- trust(objfun = penmlogl, tout$argument, rinit = 1, rmax = 10,
    sigma = oout$par, fixed = modmat.fix,
    random = list(modmat.blk, modmat.pop), obj = aout, fterm = 0)
tout$converged

sigma.better <- oout$par
alpha.better <- tout$argument[1:nfix]
c.better <- tout$argument[- (1:nfix)]
zwz.better <- makezwz(sigma.better, parm = c(alpha.better, c.better),
    fixed = modmat.fix, random = list(modmat.blk, modmat.pop), obj = aout)

### get better estimates

objfun <- function(alphaceesigma, zwz)
    pickle3(alphaceesigma, fixed = modmat.fix,
    random = list(modmat.blk, modmat.pop), obj = aout, zwz = zwz, deriv = 2)
tout <- trust(objfun, c(alpha.better, c.better, sigma.better),
    rinit = 1, rmax = 10, zwz = zwz.better)
tout$converged
alpha.mle <- tout$argument[1:nfix]
c.mle <- tout$argument[nfix + 1:(nblk + npop)]
sigma.mle <- tout$argument[nfix + nblk + npop + 1:2]
zwz.mle <- makezwz(sigma.mle, parm = c(alpha.mle, c.mle),
    fixed = modmat.fix, random = list(modmat.blk, modmat.pop), obj = aout)
```

```
### estimated variance components
sigma.mle^2

### preceding step can be iterated "until convergence"

### get (approximate) Fisher information

objfun <- function(alphasigma) pickle2(alphasigma, parm = c.mle,
    fixed = modmat.fix, random = list(modmat.blk, modmat.pop),
    obj = aout, zwz = zwz.mle)$value
gradfun <- function(alphasigma) pickle2(alphasigma, parm = c.mle,
    fixed = modmat.fix, random = list(modmat.blk, modmat.pop),
    obj = aout, zwz = zwz.mle, deriv = 1)$gradient
oout <- optim(c(alpha.mle, sigma.mle), objfun, gradfun, method = "BFGS",
    hessian = TRUE)
oout$convergence == 0
fish <- oout$hessian
```

---

predict.aster	*Predict Method for Aster Model Fits*

Description

Obtains predictions (parameter estimates) and optionally estimates standard errors of those predictions (parameter estimates) 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"),
    is.always.parameter = FALSE,
    se.fit = FALSE, info = c("expected", "observed"),
    info.tol = sqrt(.Machine$double.eps), newcoef = NULL,
    gradient = se.fit, ...)

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

Arguments

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

model.type   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 parm.type) 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.

is.always.parameter

logical, default FALSE. Only affects the result when parm.type = "mean.value" and model.type = "conditional". TRUE means the conditional mean value parameter is produced. FALSE means the conditional mean values themselves are produced (which depend on data so are not parameters). See Conditional Mean Values Section below for further explanation.

se.fit   logical switch indicating if standard errors are required.

info   the type of Fisher information to use to compute standard errors.

info.tol   tolerance for eigenvalues of Fisher information. If eval is the vector of eigenvalues of the information matrix, then eval < cond.tol * max(eval) are considered zero. Hence the corresponding eigenvectors are directions of constancy or recession of the log likelihood.

newdata   optionally, a data frame in which to look for variables with which to predict. If omitted, see modmat above. See also details section below.

varvar   a variable of length nrow(newdata), typically a variable in newdata that is a factor whose levels are character strings treated as variable names. The number of variable names is nnode. Must be of the form rep(vars, each = nind) where vars is a vector of variable names. Not used if newdata is missing.

idvar   a variable of length nrow(newdata), typically a variable in newdata that indexes individuals. The number of individuals is nind. Must be of the form rep(inds, times = nnode) where inds is a vector of labels for individuals. Not used if newdata is missing.

newcoef   if not NULL, a variable of length object$coefficients and used in its place when one wants predictions at other than the fitted coefficient values.

gradient   if TRUE return the gradient (Jacobian of the transformation) matrix. This matrix has number of rows equal to the length of the fitted values and number of columns equal to the number of regression coefficients. It is the derivative matrix (matrix of partial derivatives) of the mapping from regression coefficients to whatever the predicted values are, which depends on what the arguments newdata, amat, parm.type, and model.type are.

...   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 asks for *conditional* mean value parameters, one gets them only if `is.always.parameter = TRUE` is specified. Otherwise, conditional expectations that are not parameters (because they depend on data) are produced. See Conditional Mean Values Section for more about this.

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

`fit`	Predictions
`se.fit`	Estimated standard errors
`gradient`	The gradient of the transformation from regression coefficients to predictions

If `gradient = TRUE`, a list with components

`fit`	Predictions
`gradient`	The gradient of the transformation from regression coefficients to predictions

Conditional Mean Values

Both the original aster paper (Geyer, et al., 2007) and this package are weird about conditional mean values. Equation (10) of that paper defines (using different notation from what we use here)

$$\xi_j = E(x_j | x_{p(j)})$$

where x_j are components of the response vector and $p(j)$ denotes denotes the predecessor of node j. That paper explicitly says that this is is not a parameter because it depends on the data. In fact

$$E(x_j | x_{p(j)}) = x_{p(j)} E(x_j | x_{p(j)} = 1)$$

(this is equation (3) of that paper in different notation). Thus it is weird to use a Greek letter to denote this.

There should be a conditional mean value parameter, and Geyer (2010, equation (11b)) defines it as

$$\xi_j = E(y_j | y_{p(j)} = 1)$$

(This equation only makes sense when the conditioning event $x_{p(j)} = 1$ is possible, which it is not for k-truncated arrows for $k > 0$. Then a more complicated definition must be used. By definition x_j is the sum of $x_{p(j)}$ independent and identically distributed random variables, and ξ_j is always the mean of one of those random variables.) This gives us the important relationship between conditional and unconditional mean value parameters

$$\mu_j = \xi_j \mu_{p(j)}$$

which holds for all successor nodes j. All later writings of this author use this definition of ξ as does the R package `aster2` (Geyer, 2017). This is one of six important parameterizations of an unconditional aster model (Geyer, 2010, Sections 2.7 and 2.8). The R package `aster2` uses all of them.

This function (as of version 1.0 of this package) has an argument `is.always.parameter` to switch between these two definitions in case `parm.type = "mean.value"` and `model.type = "conditional"` are specified. Then `is.always.parameter = TRUE` specifies that the latter definition of ξ is produced (which is a parameter, with all other options for `parm.type` and `model.type`). The option `is.always.parameter = FALSE` specifies that the former definition of ξ is produced (which is a conditional expectation but not a parameter) and is what this function produced in versions of this package before 1.0.

References

Geyer, C. J. (2010) A Philosophical Look at Aster Models. Technical Report No. 676. School of Statistics, University of Minnesota. https://purl.umn.edu/57163.

Geyer, C.~J. (2017). R package `aster2` (Aster Models), version 0.3. https://cran.r-project.org/package=aster2.

Geyer, C. J., Wagenius, S., and Shaw, R. G. (2007) Aster models for life history analysis. *Biometrika*, **94**, 415–426. doi:10.1093/biomet/asm030.

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 <- grepl("hdct", as.character(redata$varb))
redata <- data.frame(redata, hdct = as.integer(hdct))
level <- gsub("[0-9]", "", as.character(redata$varb))
redata <- data.frame(redata, level = as.factor(level))
aout <- aster(resp ~ varb + level : (nsloc + ewloc) + 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 <- grepl("hdct", as.character(renewdata$varb))
renewdata <- data.frame(renewdata, hdct = as.integer(hdct))
level <- gsub("[0-9]", "", as.character(renewdata$varb))
renewdata <- data.frame(renewdata, level = as.factor(level))
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(aout, 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)
```

---

quickle   *Penalized Quasi-Likelihood for Aster Models*

Description

Evaluates the objective function for approximate maximum likelihood for an aster model with random effects. Uses Laplace approximation to integrate out the random effects analytically. The "quasi" in the title is a misnomer in the context of aster models but the acronym PQL for this procedure is well-established in the generalized linear mixed models literature.

Usage

```
quickle(alphanu, bee, fixed, random, obj, y, origin, zwz, deriv = 0)
```

Arguments

alphanu	the parameter vector value where the function is evaluated, a numeric vector, see details.
bee	the random effects vector that is used as the starting point for the inner optimization, which maximizes the penalized log likelihood to find the optimal random effects vector matching alphanu.
fixed	the model matrix for fixed effects. The number of rows is nrow(obj$data). The number of columns is the number of fixed effects.
random	the model matrix or matrices for random effects. The number of rows is nrow(obj$data). The number of columns is the number of random effects in a group. Either a matrix or a list each element of which is a matrix.
obj	aster model object, the result of a call to [aster]().
y	response vector. May be omitted, in which case obj$x is used. If supplied, must be a matrix of the same dimensions as obj$x.
origin	origin of aster model. May be omitted, in which case default origin (see [aster]()) is used. If supplied, must be a matrix of the same dimensions obj$x.
zwz	A possible value of $Z^T W Z$, where Z is the model matrix for all random effects and W is the variance matrix of the response. See details. Typically constructed by the function [makezwz]().
deriv	Number of derivatives wanted, zero, one, or two.

Details

Define
$$p(\alpha, b, \nu) = m(a + M\alpha + Zb) + \tfrac{1}{2} b^T D^{-1} b + \tfrac{1}{2} \log \det[Z^T W Z D + I]$$

where m is minus the log likelihood function of a saturated aster model, where a is a known vector (the *offset vector* in the terminology of [glm]() but the *origin* in the terminology of [aster]()), where M is a known matrix, the model matrix for fixed effects (the argument fixed of this function), where Z is a known matrix, the model matrix for random effects (either the argument random of this function if it is a matrix or Reduce(cbind, random) if random is a list of matrices), where D is a diagonal matrix whose diagonal is the vector rep(nu, times = nrand) where nrand is sapply(random, ncol) when random is a list of matrices and ncol(random) when random is a matrix, where W is an arbitrary symmetric positive semidefinite matrix ($Z^T W Z$ is the argument zwz of this function), and where I is the identity matrix. Note that D is a function of ν although the notation does not explicitly indicate this.

The argument alphanu of this function is the concatenation of the parameter vectors α and ν. The argument bee of this function is a possible value of b. The length of α is the column dimension of M. The length of b is the column dimension of Z. The length of ν is the length of the argument random of this function if it is a list and is one otherwise.

Let b^* denote the minimizer of $p(\alpha, b, \nu)$ considered as a function of b for fixed α and ν, so b^* is a function of α and ν. This function evaluates

$$q(\alpha, \nu) = p(\alpha, b^*, \nu)$$

and its gradient vector and Hessian matrix (if requested). Note that b^* is a function of α and ν although the notation does not explicitly indicate this.

Value

a list with some of the following components: value, gradient, hessian, alpha, bee, nu. The first three are the requested derivatives. The second three are the corresponding parameter values: alpha and nu are the corresponding parts of the argument alphanu, the value of bee is the result of the inner optimization (b^* in the notation in details), not the argument bee of this function.

Note

Not intended for use by naive users. Use summary.reaster, which calls it.

Examples

```
data(radish)

pred <- c(0,1,2)
fam <- c(1,3,2)

rout <- reaster(resp ~ varb + fit : (Site * Region),
    list(block = ~ 0 + fit : Block, pop = ~ 0 + fit : Pop),
    pred, fam, varb, id, root, data = radish)

alpha.mle <- rout$alpha
bee.mle <- rout$b
nu.mle <- rout$sigma^2
zwz.mle <- rout$zwz
obj <- rout$obj
fixed <- rout$fixed
random <- rout$random
alphanu.mle <- c(alpha.mle, nu.mle)

qout <- quickle(alphanu.mle, bee.mle, fixed, random, obj,
    zwz = zwz.mle, deriv = 2)
```

---

radish   *Life History Data on Raphanus sativus*

Description

Data on life history traits for the invasive California wild radish *Raphanus sativus*

Usage

```
data(radish)
data(radish2)
```

Format

A data frame with records for 286 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.

Site Categorical. Experimental site where plant was grown. Two sites in this dataset.

Block Categorical. Block nested within site.

Region Categorical. Region from which individuals were obtained: northern, coastal California (N) or southern, inland California (S).

Pop Categorical. Wild population nested within region.

varbFlowering Indicator (zero or one). Shorthand for `as.numeric(radish$varb == "Flowering")`.

varbFlowers Indicator (zero or one). Shorthand for `as.numeric(radish$varb == "Flowers")`.

fit Indicator (zero or one). Shorthand for `as.numeric(radish$varb == "Fruits")`. So-called because the components of `outcome` indicated are the best surrogate of Darwinian fitness in these data.

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

Flowering Indicator (zero or one). Bernoulli, One if individual survived to produce flowers.

Flowers Integer. Zero-truncated Poisson, number of flowers observed.

Fruits Integer. Poisson, number of fruits observed.

Graphical model is

$$1 \longrightarrow \text{Flowering} \longrightarrow \text{Flowers} \longrightarrow \text{Fruits}$$

Dataset `radish2` has exactly the same structure but the estimated value of the pop variance component is zero. These data were simulated by R function raster in this package.

Source

Caroline Ridley

References

These data are a subset of data previously analyzed using fixed effect aster methods (R function `aster`) in the following.

Ridley, C. E. and Ellstrand, N. C. (2010). Rapid evolution of morphology and adaptive life history in the invasive California wild radish (*Raphanus sativus*) and the implications for management. *Evolutionary Applications*, **3**, 64–76.

These data are a subset of data previously analyzed using random effect aster methods (R function `reaster`) in the following.

Geyer, C. J., Ridley, C. E., Latta, R. G., Etterson, J. R., and Shaw, R. G. (2013) Local Adaptation and Genetic Effects on Fitness: Calculations for Exponential Family Models with Random Effects. *Annals of Applied Statistics*, **7**, 1778–1795. doi:10.1214/13AOAS653.

See Also

`pickle`

Examples

```
data(radish)
```

---

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 `ncol(theta)` determining the graph. `pred[j]` is the index of the predecessor of the node with index j unless the predecessor is a root node, in which case `pred[j] == 0`.

fam
: integer vector of length `ncol(theta)` determining the exponential family structure of the aster model. Each element is an index into the vector of family specifications given by the argument `famlist`.

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))
```

---

reaster | *Aster Models with Random Effects*

Description

Fits Aster Models with Random Effects using Laplace Approximation.

Usage

```
reaster(fixed, random, pred, fam, varvar, idvar, root,
    famlist = fam.default(), origin, data, effects, sigma, response)
```

Arguments

fixed   either a model matrix or a formula specifying response and model matrix. The
                model matrix for fixed effects.

random   either a model matrix or list of model matrices or a formula or a list of formulas
                specifying a model matrix or matrices. The model matrix or matrices for ran-
                dom effects. Each model matrix specifies the random effects for one variance
                component.

pred   an integer vector of length nnode determining the dependence graph of the aster
                model. pred[j] is the index of the predecessor of the node with index j unless
                the predecessor is a root node, in which case pred[j] == 0. See details section
                of [aster](#) for further requirements.

fam   an integer vector of length nnode determining the exponential family structure of
                the aster model. Each element is an index into the vector of family specifications
                given by the argument famlist.

varvar   a variable whose length is the row dimension of all model matrices that is a factor
                whose levels are character strings treated as variable names. The number of
                variable names is nnode. Must be of the form rep(vars, each = nind) where
                vars is a vector of variable names. Usually found in the data frame data when
                this is produced by the [reshape](#) function.

idvar   a variable whose length is the row dimension of all model matrices. The number
                of individuals is nind. Must be of the form rep(inds, times = nnode) where
                inds is a vector of labels for individuals. Usually found in the data frame data
                when this is produced by the [reshape](#) function.

root   a vector whose length is the row dimension of all model matrices. For nodes
                whose predecessors are root nodes specifies the value of the constant at that root
                node. Typically the vector having all components equal to one.

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

origin   a vector whose length is the row dimension of all model matrices. Distinguished
                point in parameter space. May be missing, in which case an unspecified default
                is provided. See details of [aster](#) for further explanation.

data   an optional data frame containing the variables in the model. If not found in
                data, the variables are taken from environment(fixed), typically the environ-
                ment from which reaster is called. Usually produced by the [reshape](#) function.
                Not needed when model matrices rather than formulas are supplied in fixed and
                random.

effects	if not missing, a vector specifying starting values for all effects, fixed and random. Length is the sum of the column dimensions of all model matrices. If supplied, the random effects part should be standardized (random effects divided by their standard deviations, like the component c of the output of this function).
sigma	if not missing, a vector specifying starting values for the square roots of the variance components. Length is the number of model matrices for random effects (the length of the list random if a list and one if random is not a list.
response	if not missing, a vector specifying the response. Length is the row dimension of all model matrices. If missing, the response is determined by the response in the formula fixed.

Details

See the help page for the function [aster](#) for specification of aster models. This function only fits unconditional aster models (those with default values of the aster function arguments type and origin.type.

The only difference between this function and the aster function is that some effects are treated as random. The unconditional canonical parameter vector of the aster model is treated as an affine function of fixed and random effects

$$\varphi = M\beta + \sum_{i=1}^{k} \sigma_i^2 Z_i b_i$$

where M and the Z_i are model matrices specified by the arguments fixed and random, where β is a vector of fixed effects and each b_i is a vector of random effects that are assumed to be (marginally) normally distributed with mean vector zero and variance matrix σ_i^2 times the identity matrix. The vectors of random effects b_i are not parameters, rather they are latent (unobservable, hypothetical) variables. The square roots of the variance components σ_i are parameters as are the components of β.

This function maximizes an approximation to the likelihood introduced by Breslow and Clayton (1993). See Geyer, et al. (2013) for details.

Value

reaster returns an object of class inheriting from "reaster". An object of class "reaster" is a list containing at least the following components:

obj	The aster object returned by a call to the [aster](#) function to fit the fixed effects model.
fixed	the model matrix for fixed effects.
random	the model matrix or matrices for random effects.
dropped	names of columns dropped from the fixed effects matrix.
sigma	approximate MLE for square roots of variance components.
nu	approximate MLE for variance components.
c	penalized likelihood estimates for the c's, which are rescaled random effects.

b	penalized likelihood estimates for the random effects.
alpha	approximate MLE for fixed effects.
zwz	ZWZ^T where Z is the model matrix for random effects and W is the Hessian matrix of minus the complete data log likelihood with respect to random effects with MLE values of the parameters plugged in.
response	the response vector.
origin	the origin (offset) vector.
iterations	number of iterations of trust region algorithm in each iteration of re-estimating zwz and re-fitting.
counts	number of iterations of Nelder-Mead in initial optimization of approximate missing data log likelihood.
deviance	up to a constant, minus twice the maximized value of the Breslow-Clayton approximation to the log-likelihood. (Note the minus. This is somewhat counter-intuitive, but agrees with the convention used by the aster function.)

Calls to reaster.formula return a list also containing:

call	the matched call.
formula	the formulas supplied.

NA Values

It was almost always wrong for aster model data to have NA values. Although theoretically possible for the R formula mini-language to do the right thing for an aster model with NA values in the data, usually it does some wrong thing. Thus, since version 0.8-20, this function and the aster function give errors when used with data having NA values. Users must remove all NA values (or replace them with what they should be, perhaps zero values) "by hand".

Warning about Negative Binomial

The negative binomial and truncated negative binomial are fundamentally incompatible with random effects. The reason is that the canonical parameter space for a one-parameter negative binomial or truncated negative binomial is the negative half line. Thus the conditional canonical parameter θ for such a node must be negative valued. The aster transform is so complicated that it is unclear what the corresponding constraint on the unconditional canonical parameter φ is, but there is a constraint: its parameter space is not the whole real line. A normal random effect, in contrast, does have support the whole real line. It wants to make parameters that are constrained to have any real number. The code only warns about this situation, because if the random effects do not influence any negative binomial or truncated negative binomial nodes of the graph, then there would be no problem.

Warning about Individual Random Effects

The Breslow-Clayton approximation assumes the complete data log likelihood is approximately quadratic considered as a function of random effects only. This will be the case by the law of large numbers if the number of individuals is much larger than the number of random effects. Thus Geyer, et al. (2013) warn against trying to put a random effect for each individual in the model. If you do that, the code will try to fit the model, but it will take forever and no theory says the results will make any sense.

References

Breslow, N. E., and Clayton, D. G. (1993). Approximate Inference in Generalized Linear Mixed Models. *Journal of the American Statistical Association*, **88**, 9–25. doi:10.1080/01621459.1993.10594284.

Geyer, C. J., Ridley, C. E., Latta, R. G., Etterson, J. R., and Shaw, R. G. (2012) Aster Models with Random Effects via Penalized Likelihood. Technical Report 692, School of Statistics, University of Minnesota. https://purl.umn.edu/135870.

Geyer, C. J., Ridley, C. E., Latta, R. G., Etterson, J. R., and Shaw, R. G. (2013) Local Adaptation and Genetic Effects on Fitness: Calculations for Exponential Family Models with Random Effects. *Annals of Applied Statistics*, **7**, 1778–1795. doi:10.1214/13AOAS653.

Examples

```
library(aster)
data(radish)
pred <- c(0,1,2)
fam <- c(1,3,2)
rout <- reaster(resp ~ varb + fit : (Site * Region),
    list(block = ~ 0 + fit : Block, pop = ~ 0 + fit : Pop),
    pred, fam, varb, id, root, data = radish)
summary(rout)
summary(rout, stand = FALSE)
```

---

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 `resp ~ 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 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. https://hdl.handle.net/11299/56204.

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. https://hdl.handle.net/11299/56219.

Examples

```
data(sim)
## Not run:
### CRAN policy says examples must take < 5 sec.  This doesn't.
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)

## End(Not run)
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

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

Directions of Recession

This function may give an error message "cannot compute standard errors, apparent directions of recession". There are two reasons why this can happen.

- There may actually be a direction of recession (DOR). Then the maximum likelihood estimate does not exist; increasing the likelihood drives (some of) the coefficients to infinity or minus infinity.

  This function's guess at the DOR can be extracted from the error object obtained by wrapping this function in try and then extracting the dor component of the condition attribute of the error object. An example of this is on the help page for the foobar data set.

  This function's guessed DOR are apparent null eigenvector(s) of the Fisher information matrix. Due to inaccuracy of computer arithmetic, this is only a guess. What are deemed null eigenvectors is controlled by the info.tol argument of this function. Reducing info.tol to perhaps 1e-9 or 1e-10 or even a little lower may make the putative DOR go away. In this case they were probably bogus (see next item). Reducing info.tol to near or below the machine epsilon .Machine$double.eps (.Machine) instructs this function to feed you garbage with no error or warning.

Putative DOR are probably true DOR if they are highly patterned with many zero or nearly zero components and other components that are nearly (small) integer multiples of each other. Putative DOR are probably bogus if they look like random noise.

DOR, if true, cannot simply be ignored. For more information, including how to do more rigorous investigation of whether putative DOR are true or bogus, see the example on the help page for the [foobar](#) data set and the reference cited on that help page.

- All of the putative directions of recession may be bogus. Due to inaccuracy of computer arithmetic, ill-conditioning of predictor variables, or ill-conditioning of the aster model itself (large graphs cause more inaccurate computation), what appear to be null eigenvectors of the Fisher information matrix need not be true null eigenvectors.

In this case, the problem will go away when info.tol is decreased slightly. Only when one has proved that there is no DOR, should one use info.tol = 1e-20 which says to ignore the problem altogether (whether putative DOR are true or bogus).

See Also

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

---

summary.reaster   *Summarizing Aster Model with Random Effects Fits*

Description

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

Usage

```
## S3 method for class 'reaster'
summary(object, standard.deviation = TRUE, ...)

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

Arguments

object   an object of class "reaster", usually, a result of a call to [reaster](#).

standard.deviation

if TRUE, treat the parameters described in the "variance components" section of the printout are square roots of variance components (that is, standard deviations) rather than the variance components themselves. Warning: if FALSE so actual variance components are described, (asymptotic, approximate) standard errors are zero when they the variance components are zero (see details section below).

x   an object of class "summary.reaster", usually, a result of a call to summary.reaster.

digits   the number of significant digits to use when printing.

signif.stars   logical. If TRUE, "significance stars" are printed for each coefficient.

...   further arguments passed to or from other methods.

Details

The [reaster](#) function only does approximate maximum likelihood. Even if it did actual maximum likelihood, standard errors would be only approximate. Standard errors for variance components are derived via the delta method from standard errors for square roots of variance components (standard deviations). Hence P-values for variance components and square roots of variance components do not agree exactly (although they do asymptotically).

Value

summary.reaster returns an object of class "summary.reaster".

See Also

[reaster](#), [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

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

---

vcov.aster   *Calculate Asymptotic Variance-Covariance Matrix for a Fitted Model*
                           Object

Description

vcov() is a generic function and functions with names beginning in vcov. will be methods for this
function.

Usage

```
## S3 method for class 'aster'
vcov(object, info = c("expected", "observed"), ...)
## S3 method for class 'reaster'
vcov(object, standard.deviation = TRUE, re.too = FALSE,
  complete = TRUE, ...)
```

Arguments

object	an object of class "aster" or class "reaster", usually the object returned by a call to R function aster or R function reaster.
info	the type of Fisher information used to compute the variance-covariance matrix.
standard.deviation	
	if FALSE, then some of the parameters are variance components, if TRUE, then the square roots of these variance components.
re.too	if TRUE, treat the estimated random effects vector, the mode of the conditional distribution of random effects given observed data, as part of the estimates vector described by the returned variance-covariance matrix. If also standard.deviation is TRUE, then these estimates are standardized.
complete	If TRUE, the full variance-covariance matrix, including entries for parameters estimated to be zero (for which the entries will also be zero).
...	required by the generic function. Ignored by these methods.

Value

A matrix of the estimated approximate, large sample, covariances between the parameter estimates.

Entries for parameters estimated to be zero are zero, not because there is no sampling variability but rather because the sampling distribution is highly non-normal and one-sided so covariances are useless.

This returned matrix has dimension attributes, like any R matrix. For the function vcov.reaster, it also has attributes, which are some of is.alpha, is.b, is.nu, is.c, and is.sigma. These are all logical vectors that can serve as index vectors for the matrix.

- is.alpha extracts elements of the variance-covariance matrix for estimates of fixed effects.
- is.nu extracts elements of the variance-covariance matrix for estimates of variance components (if standard.deviation == FALSE was specified).
- is.b extracts elements of the variance-covariance matrix for estimates of random effects (if standard.deviation == FALSE & re.too = TRUE was specified).
- is.nu extracts elements of the variance-covariance matrix for estimates of square roots of variance components (if standard.deviation == TRUE was specified).
- is.c extracts elements of the variance-covariance matrix for standardized estimates of random effects (if standard.deviation == TRUE & re.too = TRUE was specified).

See reaster for more about these parameterizations.

See Also

aster, reaster, vcov

Examples

```
data(radish2)

pred <- c(0,1,2)
fam <- c(1,3,2)

rout <- reaster(resp ~ varb + fit : (Site * Region),
    list(block = ~ 0 + fit : Block, pop = ~ 0 + fit : Pop),
    pred, fam, varb, id, root, data = radish2)

vout <- vcov(rout, standard.deviation = FALSE, re.too = TRUE)
attributes(vout)
is.alpha <- attr(vout, "is.alpha")
vout[is.alpha, is.alpha]
```

Index

58