Package 'aster2'

July 22, 2025

Version 0.3-2

Date 2024-09-17

Title Aster Models

Depends R (>= 3.6.0), Matrix

Imports stats

Suggests aster

ByteCompile TRUE

Description Aster models 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, 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). Unlike the aster package, this package does dependence groups (nodes of the graph need not be conditionally independent given their predecessor node), including multinomial and two-parameter normal as families. Thus this package also generalizes mark-capture-recapture analysis.

License GPL (>= 2)

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 2024-09-17 22:10:11 UTC

Contents

`aster2-package` *Aster Models*

Description

Aster models are exponential family graphical models that combine aspects of generalized linear models and survival analysis.

This package is still under development, only about half finished. However, it does do maximum likelihood for unconditional aster models with dependence groups, which the old package `aster` does not.

The main differences between this package and the old package are as follows.

1. The old package had triple indices for model matrices. The first index ran over individuals, the second index over nodes of the graph for an individual, and the third index over regression coefficients. Consequently the model matrix was represented (sometimes, but not consistently) as a three-dimensional array rather than a matrix, which was very confusing, even to the package author. This package ignores individuals, one index runs over all nodes of the combined graph for all individuals. Thus model matrices are always matrices.
2. The old package did not implement dependence groups, although they were described in Geyer, Wagenius and Shaw (2007). This package does. Consequently, this package requires a data frame, a vector `pred` that indicates predecessors, a vector `group` that indicates individuals in the same dependence group, and a vector `fam` that indicates families to specify a saturated aster model (the old package required only the data frame, `pred`, and `fam`). To facilitate the old style model specification, there is a new function `asterdata` that constructs objects of class `"asterdata"` given an old style data frame, `pred`, and `fam`. All other functions of the package take objects of class `"asterdata"` as model specifications.
3. The function `predict.aster` in the old package did some parameter transformations, but not all, and the returned value, when a list, had a component `gradient`, that was undocumented but useful in applying the delta method. The functions `transformSaturated`, `transformConditional`, and `transformUnconditional` in this package transform between any of the following parameter vectors: the conditional canonical parameter θ, the unconditional canonical parameter

φ, the conditional mean value parameter ξ, the unconditional mean value parameter μ, the canonical affine submodel canonical parameter β, and (unconditional aster models only) the canonical affine submodel mean value parameter τ (this last parameter is new, not discussed in the cited papers below, it is $\tau = M^T \mu$, where M is the model matrix). The change of parameter from τ to β is equivalent to maximum likelihood estimation for an unconditional aster model when the value $\tau = M^T y$ is used, where y is the response vector. All of these transformation functions also compute derivatives, if requested. See examples.

Bugs

Functions analogous to `aster`, `anova`, and `predict` in the old package are missing, thus model fitting, hypothesis tests, and confidence intervals are more cumbersome. In fact, since there is no function to calculate log likelihoods (like `mlogl` in the old package), there is no way to do likelihood ratio tests (but Rao or Wald tests could be done, for unconditional aster models, since the derivative of the log likelihood is observed minus expected $M^T(y - \mu)$.

References

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

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H. and Etterson, J. R. (2008) Unifying Life History Analyses for Inference of Fitness and Population Growth. *American Naturalist*, **172**, E35–E47.

See Also

`asterdata`, `transformSaturated`, `families`

Examples

```
## Not run: # perfectly good example but takes longer to run than CRAN allows
data(echinacea)
#### estimate MLE (simpler model than in Biometrika paper cited, not as good)
hdct <- as.numeric(grepl("hdct", as.character(echinacea$redata$varb)))
modmat <- model.matrix(resp ~ varb + nsloc + ewloc + pop * hdct - pop,
    data = echinacea$redata)
tau.hat <- as.numeric(t(modmat) %*% echinacea$redata$resp)
beta.hat <- transformUnconditional(tau.hat, modmat, echinacea,
    from = "tau", to = "beta")
inverse.fisher <- jacobian(tau.hat, echinacea, transform = "unconditional",
    from = "tau", to = "beta", modmat = modmat)
#### now have MLE (beta.hat) and pseudo-inverse of Fisher information
#### (inverse.fisher), pseudo-inverse because modmat is not full rank
foo <- cbind(beta.hat, sqrt(diag(inverse.fisher)))
foo <- cbind(foo, foo[ , 1]/foo[ , 2])
foo <- cbind(foo, 2 * pnorm(- abs(foo[ , 3])))
dimnames(foo) <- list(colnames(modmat),
    c("Estimate", "Std. Error", "z value", "Pr(>|z|)"))
printCoefmat(foo)
#### coefficients constrained to be zero because parameterization is not
#### identifiable have estimate zero and std. error zero (and rest NA)
```

```
#### estimate fitness in populations
#### generate new data with one individual in each pop at location (0, 0)
pop.names <- levels(echinacea$redata$pop)
pop.idx <- match(pop.names, as.character(echinacea$redata$pop))
pop.id <- echinacea$redata$id[pop.idx]
newdata <- subset(echinacea, echinacea$redata$id %in% pop.id)
newdata$redata[ , "nsloc"] <- 0
newdata$redata[ , "ewloc"] <- 0
hdct <- as.integer(grepl("hdct", as.character(newdata$redata$varb)))
#### modmat for new data
newmodmat <- model.matrix(resp ~ varb + nsloc + ewloc + pop * hdct - pop,
    data = newdata$redata)
#### matrix that when multiplied mean value parameter vector gives fitness
#### in each pop
amat <- matrix(NA, nrow = length(pop.id), ncol = nrow(newmodmat))
for (i in 1:nrow(amat))
    amat[i, ] <- as.numeric(grepl(paste("^", pop.id[i], ".hdct", sep = ""),
        rownames(newmodmat)))
#### transform to expected fitness parameters
efit <- transformUnconditional(beta.hat, newmodmat, newdata,
    from = "beta", to = "mu")
efit <- as.numeric(amat %*% efit)
#### jacobian matrix of this transformation
jack <- jacobian(beta.hat, newdata, transform = "unconditional",
    from = "beta", to = "mu", modmat = newmodmat)
#### delta method standard errors
sefit <- sqrt(diag(amat %*% jack %*% inverse.fisher %*% t(jack) %*% t(amat)))
foo <- cbind(efit, sefit)
dimnames(foo) <- list(pop.names, c("Est. fitness", "Std. Error"))
print(foo)

## End(Not run)
```

`asterdata` *Object Describing Saturated Aster Model*

Description

Functions to construct and test conformance to the contract for objects of class `"asterdata"`. All other functions in this package take model descriptions of this form.

Usage

```
asterdata(data, vars, pred, group, code, families, delta,
  response.name = "resp", varb.name = "varb",
  tolerance = 8 * .Machine$double.eps)
validasterdata(object, tolerance = 8 * .Machine$double.eps)
is.validasterdata(object, tolerance = 8 * .Machine$double.eps)
```

Arguments

`data`	a data frame containing response and predictor variables for the aster model.
`vars`	a character vector containing names of variables in the data frame `data` that are components of the response vector of the aster model.
`pred`	an integer vector satisfying `length(pred) == length(vars)` specifying the arrows of the subgraph of the aster model corresponding to a single individual. Must be nonnegative and satisfy `all(pred < seq(along = pred))`. A zero value of `pred[j]` indicates the predecessor of node `j` is an initial node (formerly called root node) of the subgraph. A nonzero value of `pred[j]` indicates the predecessor of node `j` is node `pred[j]`. In either case there is an arrow in the subgraph from predecessor node to successor node.
`group`	an integer vector satisfying `length(group) == length(vars)` specifying the lines of the subgraph of the aster model corresponding to a single individual, which in turn specify the dependence groups. Must be nonnegative and satisfy `all(group < seq(along = group))`. Nonzero elements of `group` indicate nodes of the subgraph that are connected by a line and hence are in the same dependence group: nodes `j` and `group[j]` are connected by a line. Since nodes in the same dependence group must have the same predecessor, this requires `pred[group[j]] == pred[j]`. Since nodes in the same dependence group must be in the same family, this requires `code[group[j]] == code[j]`. It also requires that the dimension of the family specified by `code[j]` be the same as the number of nodes in the dependence group. Zero elements of `group` indicate nothing about dependence groups. The lines indicate a transitive relation. If there is a line from node `j1` to node `j2` and a line from node `j2` to node `j3` then there is also a line from node `j1` to node `j3`, but this line need not be specified by the `group` vector, and indeed cannot. If there is a dependence group with `d` nodes, then there are `choose(d, 2)` lines connecting these nodes, but the `group` vector can only specify `d - 1` lines which imply the rest. For example, if nodes `j1`, `j2`, `j3`, and `j4` are to make up a four-dimensional dependence group and `j1 < j2`, `j2 < j3`, and `j3 < j4`, we must have `group[j1] == 0`, `group[j2] == j1`, `group[j3] == j2`, and `group[j4] == j3`. This is forced by the requirement `all(group < seq(along = group))`.
`code`	an integer vector satisfying `length(code) == length(vars)` specifying the families corresponding to the dependence groups. This requires `all(code %in% seq(along = families)` Node `j` is in a dependence group with family described by `families[code[j]]`. Note that `group[j] == k` requires `families[j] == families[k]` when `k != 0`.
`families`	a list of family specifications (see `families`). Specifications of families not having hyperparameters may be abbreviated as character strings, for example, `"binomial"` rather than `fam.binomial()`.
`delta`	a numeric vector satisfying `length(delta) == length(vars)` specifying the degeneracies of the aster model for a single individual. The model specified is the limit as $s \to \infty$ of nondegenerate models having conditional canonical parameter vector $\theta + s\delta$ (note that the conditional canonical parameter vector is

always used here, regardless of whether conditional or unconditional canonical affine submodels are to be used). May be missing (and usually is) in which case $\delta = 0$ is implied, meaning the limit is trivial (same as not taking a limit).

`response.name` a character string giving the name of the response vector.

`varb.name` a character string giving the name of the factor covariate that says which of the variables in the data frame `data` correspond to which components of the response vector.

`tolerance` numeric >= 0. Relative errors smaller than `tolerance` are not considered in checking validity of normal location-scale data.

`object` an object of class `"asterdata"`. The function `validasterdata` always returns `TRUE` or throws an error with an informative message. The function `is.validasterdata` never throws an error unless `object` has the wrong class, returning `TRUE` or `FALSE` according to whether `object` does or does not conform to the contract for class `"asterdata"`.

Details

Response variables in dependence groups are taken to be in the order they appear in the response vector. The first to appear in the response vector is the first canonical statistic for the dependence group distribution, the second to appear the second canonical statistic, and so forth. The number of response variables in the dependence group must match the dimension of the dependence group distribution.

This function only handles the usual case where the subgraph for every individual is isomorphic to subgraph for every other individual and all initial nodes (formerly called root nodes) correspond to the constant one. Each row of `data` is the data for one individual. The vectors `vars`, `pred`, `group`, `code`, and `delta` (if not missing) describe the subgraph for one individual (which is the same for all individuals).

In other cases for which this function does not have the flexibility to construct the appropriate object of class `"asterdata"`, such an object will have to be constructed "by hand" using R statements not involving this function or modifying an object produced by this function. See the following section for description of such objects. The functions `validasterdata` and `is.validasterdata` can be used to check whether objects constructed "by hand" have been constructed correctly.

Value

an object of class `"asterdata"` is a list containing the following components

`redata` a data frame having `nrow(data) * length(vars)` rows and containing variables having names in `setdiff(names(data), vars)` and also the names `"id"`, `response.name`, and `varb.name`. Produced from `data` using the `reshape` function. Each variable in `setdiff(names(data), vars)` is repeated `length(vars)` times. The variable named `response.name` is the concatenation of the variables in `data` with names in `vars`. The variable named `varb.name` is a factor having levels `vars` that says which of the variables in the data frame `data` correspond to which components of the response vector. The variable named `"id"` is an integer vector that says which of the individuals (which rows of `data`) correspond to which rows of `redata`. Not all objects of class `"asterdata"` need have an `id` variable, although all those constructed by this function do.

repred — an integer vector satisfying `length(repred) == nrow(redata)` specifying the arrows of the graph of the aster model for all individuals. Must be nonnegative and satisfy `all(repred < seq(along = repred))`. A zero value of `repred[j]` indicates the predecessor of node `j` is an initial node (formerly called root node) of the graph. A nonzero value of `repred[j]` indicates the predecessor of node `j` is node `repred[j]`. In either case there is an arrow in the graph from predecessor node to successor node.

Note that `repred` is determined by `pred` but is quite different from it. Firstly, the lengths differ. Secondly, `repred` is not just a repetition of `pred`. The numbers in `pred`, if nonzero, are indices for the vector `vars` whereas the numbers in `repred`, if nonzero, are row indices for the data frame `redata`.

initial — a numeric vector specifying constants associated with initial nodes (formerly called root nodes) of the graphical model for all individuals. If `repred[j] == 0` then the predecessor of node `j` is an initial node associated with the constant `initial[j]`, which must be a positive integer unless the family associated with the arrow from this initial node to node `j` is infinitely divisible (the only such family currently implemented being Poisson), in which case `initial[j]` must be a strictly positive and finite real number. If `repred[j] != 0`, then `initial[j]` is ignored and may be any numeric value, including `NA` or `NaN`. This function always makes `initial` equal to `rep(1, nrow(redata))` but the more general description above is valid for objects of class `"asterdata"` constructed "by hand".

regroup — an integer vector satisfying `length(regroup) == nrow(redata)` specifying the lines of the graph of the aster model for all individuals, which in turn specify the dependence groups. Must be nonnegative and satisfy `all(regroup < seq(along = regroup))`. Nonzero elements of `regroup` indicate nodes of the graph that are connected by a line and hence are in the same dependence group: nodes `j` and `regroup[j]` are connected by a line. Since nodes in the same dependence group must have the same predecessor, this requires `repred[regroup[j]] == repred[j]`. Since nodes in the same dependence group must be in the same family, this requires `recode[regroup[j]] == recode[j]`.

It also requires that the dimension of the family specified by `recode[j]` be the same as the number of nodes in the dependence group. Zero elements of `regroup` indicate nothing about dependence groups.

The lines indicate a transitive relation. If there is a line from node `j1` to node `j2` and a line from node `j2` to node `j3` then there is also a line from node `j1` to node `j3`, but this line need not be specified by the `group` vector, and indeed cannot. If there is a dependence group with `d` nodes, then there are `choose(d, 2)` lines connecting these nodes, but the `group` vector can only specify `d - 1` lines which imply the rest.

For example, if nodes `j1`, `j2`, `j3`, and `j4` are to make up a four-dimensional dependence group and `j1 < j2`, `j2 < j3`, and `j3 < j4`, we must have `regroup[j1] == 0`, `regroup[j2] == j1`, `regroup[j3] == j2`, and `regroup[j4] == j3`. This is forced by the requirement `all(regroup < seq(along = regroup))`.

Note that `regroup` is determined by `group` but is quite different from it. Firstly, the lengths differ. Secondly, `regroup` is not just a repetition of `group`. The numbers in `group`, if nonzero, are indices for the vector `vars` whereas the numbers in `regroup`, if nonzero, are row indices for the data frame `redata`.

`recode` an integer vector satisfying `length(recode) == nrow(redata)` specifying the families corresponding to the dependence groups. This requires

```
all(recode %in% seq(along = families)
```

Node `j` is in a dependence group with family described by `families[recode[j]]`.

Note that `regroup[j] == k` requires `recode[j] == recode[k]` when `regroup[j] != 0`. Also note that `recode` is determined by `code` but is different from it. Firstly, the lengths differ. Secondly, `recode` need not be just a repetition of `code`. This function always makes `recode` equal to `rep(code, each = nrow(redata))` but the more general description above is valid for objects of class `"asterdata"` constructed "by hand".

`families` a copy of the argument of the same name of this function except that any character string abbreviations are converted to objects of class `"astfam"`.

`redelta` a numeric vector satisfying `length(redelta) == nrow(redata)` specifying the degeneracies of the aster model for all individuals. If not the zero vector, the degenerate model specified is the limit as $s \to \infty$ of nondegenerate models having conditional canonical parameter vector $\theta + s\delta$ (note that the conditional canonical parameter vector is always used here, regardless of whether conditional or unconditional canonical affine submodels are to be used).

Note that `redelta` is determined by `delta` but is different from it. Firstly, the lengths differ. Secondly, `redelta` need not be just a repetition of `delta`. This function always makes `redelta` equal to `rep(delta, each = nrow(redata))` but the more general description above is valid for objects of class `"asterdata"` constructed "by hand".

`response.name` a character string giving the name of the response variable in `redata`. For this function, a copy of the argument `response.name`.

`varb.name` a character string giving the name of the "varb" variable in `redata`. For this function, a copy of the argument `varb.name`.

In addition an object of class `"asterdata"` may contain (and those constructed by this function do contain) components `pred`, `group`, and `code`, which are copies of the arguments of the same names of this function. Objects of class `"asterdata"` not constructed by this function need not contain these additional components, since they may make no sense if the graph for all individuals is not the repetition of isomorphic subgraphs, one for each individual.

See Also

`families` and `subset.asterdata`

Examples

```
data(test1)
fred <- asterdata(test1, vars = c("m1", "n1", "n2"), pred = c(0, 1, 1),
    group = c(0, 0, 2), code = c(1, 2, 2),
    families = list("bernoulli", "normal.location.scale"))
is.validasterdata(fred)
```

constancy *Constancy Spaces for Aster Models*

Description

Produce basis for constancy space of an aster model. Test whether the difference of two canonical parameter vectors is in the constancy space (so the two parameter vectors correspond to the same probability model).

Usage

```
constancy(data, parm.type = c("theta", "phi"))
is.same(parm1, parm2, data, parm.type = c("theta", "phi"),
    tolerance = sqrt(.Machine$double.eps))
```

Arguments

data	an object of class "asterdata" produced by asterdata or "by hand" such that is.validasterdata(data) returns TRUE. The specification of the aster model.
parm.type	the parametrization for which the constancy space is wanted.
parm1	a parameter vector of the type specified by parm.type for the saturated aster model specified by data.
parm2	another parameter vector of the type specified by parm.type for the saturated aster model specified by data.
tolerance	numeric >= 0. Relative errors smaller than tolerance are not considered in the comparison.

Details

There is no need for functions to test whether different mean value parameters (ξ or μ) correspond to the same probability distribution because these parametrizations are identifiable (different valid parameter vectors correspond to different probability distributions).

Value

for is.same a logical value; for constancy a matrix whose rows constitute a basis for the constancy space. This means that if δ is a linear combination of rows of this matrix then for all real s the distributions having parameter vectors ψ and $\psi + s\delta$ are the same, where $\psi = \theta$ or $\psi = \varphi$ depending on whether parm.type = "theta" or parm.type = "phi". Conversely, if ψ_1 and ψ_2 are valid parameter vectors of the same type, then they correspond to the same probability distribution only if $\psi_1 - \psi_2$ is a linear combination of rows of this matrix.

See Also

asterdata

Examples

```
data(test1)
fred <- asterdata(test1,
    vars = c("m1", "m2", "m3", "n1", "n2", "b1", "p1", "z1"),
    pred = c(0, 0, 0, 1, 1, 2, 3, 6), group = c(0, 1, 2, 0, 4, 0, 0, 0),
    code = c(1, 1, 1, 2, 2, 3, 4, 5),
    families = list(fam.multinomial(3), "normal.location.scale",
    "bernoulli", "poisson", "zero.truncated.poisson"))
cmat <- constancy(fred, parm.type = "phi")
```

`cumulant` *Cumulant Functions for Aster Models*

Description

Calculate cumulant function and up to three derivatives for families known to the package.

Usage

```
cumulant(theta, fam, deriv = 0, delta)
```

Arguments

`theta`	canonical parameter value.
`fam`	an object of class `"astfam"` produced by one of the family functions (see `families`) specifying the exponential family.
`deriv`	the number of derivatives wanted. Must be nonnegative integer less than or equal to three.
`delta`	direction in which limit is taken. Cumulant function is for family that is limit of family specified, limit being for distributions with parameter $\theta + s\delta$, the limit being as $s \to \infty$. May be missing, in which case $\delta = 0$ is assumed, which is the same as no limit being taken.

Value

a list containing some of the following components:

`zeroth`	the value of the cumulant function at θ.
`first`	the value of the first derivative at θ, a vector having the same dimension as θ.
`second`	the value of the second derivative at θ, a $d \times d$ matrix if d is the dimension of θ or a scalar if θ is scalar.
`third`	the value of the third derivative at θ, a $d \times d \times d$ array if d is the dimension of θ or a scalar if θ is scalar.

Note

Not intended for use by ordinary users. Provides R interface for testing to C code called by many other functions in the package.

See Also

families

Examples

```
cumulant(-0.5, fam.bernoulli(), deriv = 3)
cumulant(-0.5, fam.bernoulli(), deriv = 3, delta = 1)
```

---

echinacea *Life History Data on Echinacea angustifolia*

Description

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

Usage

```
data(echinacea)
```

Format

An object of class "asterdata" (see asterdata) comprising records for 570 plants observed over three years. Nodes of the graph for one individual are associated with the variables (levels of the factor echinacea$redata$varb)

ld02 Indicator of being alive in 2002. Bernoulli, predecessor the constant one.

ld03 Ditto for 2003. Bernoulli, predecessor ld02.

ld04 Ditto for 2004. Bernoulli, predecessor ld03.

fl02 Indicator of flowering 2002. Bernoulli, predecessor ld02.

fl03 Ditto for 2003. Bernoulli, predecessor ld03.

fl04 Ditto for 2004. Bernoulli, predecessor ld04.

hdct02 Count of number of flower heads in 2002. Zero-truncated Poisson, predecessor fl02.

hdct03 Ditto for 2003. Zero-truncated Poisson, predecessor fl03.

hdct04 Ditto for 2004. Zero-truncated Poisson, predecessor fl04.

Covariates are

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.

Details

This is the data for the example in Geyer, Wagenius, and Shaw (2007). These data were included in the R package `aster` which was the predecessor of this package as the dataset `echinacea`.

Source

Stuart Wagenius, https://www.chicagobotanic.org/research/staff/wagenius

References

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

Examples

```
data(echinacea)
names(echinacea)
names(echinacea$redata)
levels(echinacea$redata$varb)
```

`families` *Families for Aster Models*

Description

Families known to the package. These functions construct simple family specifications used in specifying aster models. Statistical properties of these families are described.

Usage

```
fam.bernoulli()
fam.poisson()
fam.zero.truncated.poisson()
fam.normal.location.scale()
fam.multinomial(dimension)
```

Arguments

`dimension`	the dimension (number of categories) for the multinomial distribution.

Details

Currently implemented families are

`"bernoulli"` Bernoulli (binomial with sample size one). The distribution of any zero-or-one-valued random variable Y, which is the canonical statistic. The mean value parameter is

$$\mu = E(Y) = \Pr(Y = 1).$$

The canonical parameter is $\theta = \log(\mu) - \log(1 - \mu)$, also called logit of μ. The cumulant function is

$$c(\theta) = \log(1 + e^\theta).$$

This distribution has degenerate limiting distributions. The lower limit as $\theta \to -\infty$ is the distribution concentrated at zero, having cumulant function which is the constant function everywhere equal to zero. The upper limit as $\theta \to +\infty$ is the distribution concentrated at one, having cumulant function which is the identity function satisfying $c(\theta) = \theta$ for all θ.

For predecessor (sample size) n, the successor is the sum of n independent and identically distributed (IID) Bernoulli random variables, that is, binomial with sample size n. The mean value parameter is n times the mean value parameter for sample size one; the cumulant function is n times the cumulant function for sample size one; the canonical parameter is the same for all sample sizes.

`"poisson"` Poisson. The mean value parameter μ is the mean of the Poisson distribution. The canonical parameter is $\theta = \log(\mu)$. The cumulant function is

$$c(\theta) = e^\theta.$$

This distribution has a degenerate limiting distribution. The lower limit as $\theta \to -\infty$ is the distribution concentrated at zero, having cumulant function which is the constant function everywhere equal to zero. There is no upper limit because the canonical statistic is unbounded above.

For predecessor (sample size) n, the successor is the sum of n IID Poisson random variables, that is, Poisson with mean $n\mu$. The mean value parameter is n times the mean value parameter for sample size one; the cumulant function is n times the cumulant function for sample size one; the canonical parameter is the same for all sample sizes.

`"zero.truncated.poisson"` Poisson conditioned on being greater than zero. Let m be the mean of the corresponding untruncated Poisson distribution. Then the canonical parameters for both truncated and untruncated distributions are the same $\theta = \log(m)$. The mean value parameter for the zero-truncated Poisson distribution is

$$\mu = \frac{m}{1 - e^{-m}}$$

and the cumulant function is

$$c(\theta) = m + \log(1 - e^{-m}),$$

where m is as defined above, so $m = e^\theta$.

This distribution has a degenerate limiting distribution. The lower limit as $\theta \to -\infty$ is the distribution concentrated at one, having cumulant function which is the identity function satisfying $c(\theta) = \theta$ for all θ. There is no upper limit because the canonical statistic is unbounded above.

For predecessor (sample size) n, the successor is the sum of n IID zero-truncated Poisson random variables, which is not a brand-name distribution. The mean value parameter is n times the mean value parameter for sample size one; the cumulant function is n times the cumulant function for sample size one; the canonical parameter is the same for all sample sizes.

`"normal.location.scale"` The distribution of a normal random variable X with unknown mean m and unknown variance v. Thought of as an exponential family, this is a two-parameter family, hence must have a two-dimensional canonical statistic $Y = (X, X^2)$. The canonical parameter vector θ has components

$$\theta_1 = \frac{m}{v}$$

and

$$\theta_2 = -\frac{1}{2v}.$$

The value of θ_1 is unrestricted, but θ_2 must be strictly negative. The mean value parameter vector μ has components

$$\mu_1 = m = -\frac{\theta_1}{2\theta_2}$$

and

$$\mu_2 = v + m^2 = -\frac{1}{2\theta_2} + \frac{\theta_1^2}{4\theta_2^2}.$$

The cumulant function is

$$c(\theta) = -\frac{\theta_1^2}{4\theta_2} + \frac{1}{2}\log\left(-\frac{1}{2\theta_2}\right).$$

This distribution has no degenerate limiting distributions, because the canonical statistic is a continuous random vector so the boundary of its support has probability zero.

For predecessor (sample size) n, the successor is the sum of n IID random vectors (X_i, X_i^2), where each X_i is normal with mean m and variance v, and this is not a brand-name multivariate distribution (the first component of the sum is normal, the second component noncentral chi-square, and the components are not independent). The mean value parameter vector is n times the mean value parameter vector for sample size one; the cumulant function is n times the cumulant function for sample size one; the canonical parameter vector is the same for all sample sizes.

`"multinomial"` Multinomial with sample size one. The distribution of any random vector Y having all components zero except for one component which is one (Y is the canonical statistic vector). The mean value parameter is the vector $\mu = E(Y)$ having components

$$\mu_i = E(Y_i) = \Pr(Y_i = 1).$$

The mean value parameter vector μ is given as a function of the canonical parameter vector θ by

$$\mu_i = \frac{e^{\theta_i}}{\sum_{j=1}^{d} e^{\theta_j}},$$

where d is the dimension of Y and θ and μ. This transformation is not one-to-one; adding the same number to each component of θ does not change the value of μ. The cumulant function is

$$c(\theta) = \log\left(\sum_{j=1}^{d} e^{\theta_j}\right).$$

This distribution is degenerate. The sum of the components of the canonical statistic is equal to one with probability one, which implies the nonidentifiability of the d-dimensional canonical

parameter vector mentioned above. Hence one parameter (at least) is always constrained to to be zero in fitting an aster model with a multinomial family.

This distribution has many degenerate distributions. For any vector δ the limit of distributions having canonical parameter vectors $\theta + s\delta$ as $s \to \infty$ exists and is another multinomial distribution (the limit distribution in the direction δ). Let A be the set of i such that $\delta_i = \max(\delta)$, where $\max(\delta)$ denotes the maximum over the components of δ. Then the limit distribution in the direction δ has components Y_i of the canonical statistic for $i \notin A$ concentrated at zero. The cumulant function of this degenerate distribution is

$$c(\theta) = \log\left(\sum_{j \in A} e^{\theta_j}\right).$$

The canonical parameters θ_j for $j \notin A$ are not identifiable, and one other canonical parameter is not identifiable because of the constraint that the sum of the components of the canonical statistic is equal to one with probability one.

For predecessor (sample size) n, the successor is the sum of n IID multinomial-sample-size-one random vectors, that is, multinomial with sample size n. The mean value parameter is n times the mean value parameter for sample size one; the cumulant function is n times the cumulant function for sample size one; the canonical parameter is the same for all sample sizes.

Value

a list of class `"astfam"` giving name and values of any hyperparameters.

Examples

```
fam.bernoulli()
fam.multinomial(4)
```

`hornworm` *Life History Data on Manduca sexta*

Description

Data on life history traits for the tobacco hornworm *Manduca sexta*

Usage

```
data(hornworm)
```

Format

An object of class `"asterdata"` (see `asterdata`) comprising records for 162 insects (54 female, 68 male, and 40 for which there was no opportunity to determine sex) observed over 40 days. Nodes of the graph for one individual are associated with the variables (levels of the factor `hornworm$redata$varb`) in dependence groups

P Bernoulli. Predecessor 1 (initial node). Indicator of pupation.

T330, T331, T332 Three-dimensional multinomial dependence group. Predecessor `P`.

- **T330** Indicator of death after pupation. In these data, all deaths after pupation are considered to have happened on day 33 regardless of when they occurred (because the actual day of death was not recorded in the original data).
- **T331** Indicator of survival to day 33 but still pre-eclosion.
- **T332** Indicator of eclosion (emergence from pupa as adult moth on day 33.

B33 Zero-truncated Poisson. Predecessor `T332`. Count of ovarioles on day 33. Only females have this node in their graphs.

Tx1, Tx2 For `x` = 34, ..., 40. Two-dimensional multinomial dependence group. Predecessor `Tw1`, where `w = x - 1`.

- **Tx1** Indicator of survival to day `x` but still pre-eclosion.
- **Tx2** Indicator of eclosion (emergence from pupa as adult moth on day `x`.

Bx Zero-truncated Poisson. Predecessor `Tx2`. Count of ovarioles on day `x`. Only females have these nodes in their graph.

Covariates are

Sex a factor. `F` is known female, `M` is known male, `U` is unknown (no opportunity to observe).

Time_2nd time (in weeks) to reach the 2nd instar stage. Larval instars are stages between molts (shedding of exoskeleton) of the larval form (caterpillar).

Mass_2nd mass (in grams) at the 2nd instar stage.

Mass_Repro mass (in grams) at eclosion.

LarvaID name of an individual in the original data.

Details

This is the data described by and analyzed by non-aster methods by Kingsolver et al. (2012) and re-analyzed using this package by Eck et al. (submitted).

For an illustration of the graph, see Figure 1 in Eck et al. (submitted).

In the description above, a concrete example of the `x` and `w` notation is that T351 and T352 form a two-dimensional multinomial dependence group, the predecessor of which is T341, and B35 is a dependence group all by itself, its predecessor being T352.

Every multinomial dependence group acts like a switch. If the predecessor is one, the dependence group is multinomial with sample size one (exactly one variable is one and the rest are zero). So this indicates which way the life history goes. If the predecessor is zero, then all successors are zero. This goes for all variables in any aster model. If `Tx2` is zero, then so is `Bx`. The ovariole count is zero except for the day on which the individual eclosed.

Source

Joel Kingsolver https://bio.unc.edu/people/faculty/kingsolver/

References

Kingsolver, J. G., Diamond, S. E., Seiter, S. A., and Higgins, J. K. (2012) Direct and indirect phenotypic selection on developmental trajectories in *Manduca sexta*. *Functional Ecology* **26** 598–607.

Eck, D., Shaw, R. G., Geyer, C. J., and Kingsolver, J. (submitted) An integrated analysis of phenotypic selection on insect body size and development time.

Examples

```
data(hornworm)
names(hornworm)
names(hornworm$redata)
levels(hornworm$redata$varb)
```

`link` *Link Functions for Aster Models*

Description

Calculate link function and up to one derivative for families known to the package.

Usage

```
link(xi, fam, deriv = 0, delta)
```

Arguments

`xi`	mean value parameter value, a numeric vector.
`fam`	an object of class `"astfam"` produced by one of the family functions (see `families`) specifying the exponential family.
`deriv`	the number of derivatives wanted. Must be either zero or one.
`delta`	direction in which limit is taken. Link function is for family that is limit of family specified, limit being for distributions with canonical parameter $\theta + s\delta$, the limit being as $s \to \infty$. May be missing, in which case $\delta = 0$ is assumed, which is the same as no limit being taken.

Value

a list containing some of the following components:

`zeroth` the value of the link function at ξ, a vector of dimension d, where d

is the dimension of ξ.

`first` the value of the first derivative at ξ, a $d \times d$ matrix, where d is the dimension of ξ or a scalar if ξ is scalar.

Note

Not intended for use by ordinary users. Provides R interface for testing to C code called by many other functions in the package.

See Also

families and cumulant

Examples

```
link(0.3, fam.bernoulli(), deriv = 1)
link(0.3, fam.bernoulli(), deriv = 1, delta = 1)
```

---

subset.asterdata *Subset Object Describing Saturated Aster Model*

Description

Subset an object of class "asterdata", for which see asterdata.

Usage

```
## S3 method for class 'asterdata'
subset(x, subset, successors = TRUE, ...)
```

Arguments

x	an object of class "asterdata", for which see asterdata.
subset	a logical vector indicating nodes of the graph to keep: missing values are taken as false.
successors	a logical scalar indicating whether the subgraph must be a union of connected components of the original graph, that is, if all successors of nodes in the subset must also be in the subset.
...	further arguments, which are ignored (this argument is required for methods of the generic function subset but is not used for this method.)

Details

Argument subset is a logical vector of the same length as the number of nodes in the graph specified by argument x. It indicates the subset of nodes in the subgraph wanted. The subgraph must be closed with respect to predecessors (all predecessors of nodes in the subset are also in the subset) and if successors = TRUE with respect to successors (all successors of nodes in the subset are also in the subset). And similarly for dependence groups: each dependence group in the original graph must have all or none of its elements in the subgraph.

Value

an object of class "asterdata" that represents the aster model having subgraph with nodes specified by subset.

See Also

asterdata

Examples

```
data(echinacea)
#### select one individual from each level of pop
foo <- echinacea$redata$pop
bar <- match(levels(foo), as.character(foo))
baz <- is.element(echinacea$redata$id, echinacea$redata$id[bar])
out <- subset(echinacea, baz)
```

---

test1 *Test Data*

Description

Test data of no biological interest. Does have all families implemented at the time the test data was created. No predictor variables.

Usage

```
data(test1)
```

Format

A data frame with 100 observations on the following 8 variables.

- m1 a numeric vector, part of a multinomial dependence group (with m2 and m3). Predecessor of this group is the constant 1.
- m2 a numeric vector.
- m3 a numeric vector.
- n1 a numeric vector, part of a normal location-scale dependence group (with n2). Predecessor of this group is m1.
- n2 a numeric vector (actually n1^2).
- b1 a numeric vector, Bernoulli. Predecessor is m2.
- p1 a numeric vector, Poisson. Predecessor is m3.
- z1 a numeric vector, zero-truncated Poisson. Predecessor is b1.

Source

created by R script test1.R in directory makedata of the installation directory for this package.

Examples

```
data(test1)
fred <- asterdata(test1,
    vars = c("m1", "m2", "m3", "n1", "n2", "b1", "p1", "z1"),
    pred = c(0, 0, 0, 1, 1, 2, 3, 6), group = c(0, 1, 2, 0, 4, 0, 0, 0),
    code = c(1, 1, 1, 2, 2, 3, 4, 5),
    families = list(fam.multinomial(3), "normal.location.scale",
    "bernoulli", "poisson", "zero.truncated.poisson"))
```

Transform *Change-of-Parameter Functions for Aster Models*

Description

Calculate a change-of-parameter for an aster model or the derivative of such a change-of-parameter. Validate certain parameter vectors.

Usage

```
transformSaturated(parm, data, from = c("theta", "phi", "xi", "mu"),
  to = c("theta", "phi", "xi", "mu"), differential,
  model.type = c("unconditional", "conditional"),
  tolerance = 8 * .Machine$double.eps)
transformConditional(parm, modmat, data, from = "beta",
  to = c("theta", "phi", "xi", "mu"), differential,
  offset, tolerance = 8 * .Machine$double.eps)
transformUnconditional(parm, modmat, data, from = c("beta", "tau"),
  to = c("beta", "theta", "phi", "xi", "mu", "tau"),
  differential, offset, tolerance = 8 * .Machine$double.eps)
jacobian(parm, data,
    transform = c("saturated", "conditional", "unconditional"),
    from = c("beta", "theta", "phi", "xi", "mu", "tau"),
    to = c("beta", "theta", "phi", "xi", "mu", "tau"),
    modmat, offset, tolerance = 8 * .Machine$double.eps)
validtheta(data, theta, model.type = c("unconditional", "conditional"),
    tolerance = 8 * .Machine$double.eps)
is.validtheta(data, theta, model.type = c("unconditional", "conditional"),
    tolerance = 8 * .Machine$double.eps)
validxi(data, xi, model.type = c("unconditional", "conditional"),
    tolerance = 8 * .Machine$double.eps)
is.validxi(data, xi, model.type = c("unconditional", "conditional"),
    tolerance = 8 * .Machine$double.eps)
```

Arguments

`parm`	parameter vector to transform, a numerical vector of length `nrow(data$redata)` for `transformSaturated` or of length `ncol(modmat)` for `transformConditional` and `transformUnconditional`.

data	an object of class "asterdata" produced by asterdata or "by hand" such that is.validasterdata(data) returns TRUE. The specification of the aster model.
from	the kind of parameter which parm is. May be abbreviated.
to	the kind of parameter to which parm is to be converted. May be abbreviated.
differential	if not missing a numeric vector of the same length as parm. If missing the change-of-parameter is calculated. If not missing the directional derivative of the change-of-parameter is calculated (see Details section).
modmat	the model matrix for a canonical affine submodel, a numerical matrix having nrow(data$redata) rows and length(beta) columns for transformConditional or length(parm) columns for transformUnconditional.
offset	the offset vector for a canonical affine submodel, a numerical vector of length nrow(data$redata). May be missing, in which case offset vector equal to zero is used.
theta	conditional canonical parameter vector to validate, a numerical vector of length nrow(data$redata).
xi	conditional canonical parameter vector to validate, a numerical vector of length nrow(data$redata).
model.type	which kind of model (see Details section). May be abbreviated.
tolerance	numeric >= 0. Relative errors smaller than tolerance are not considered in checking validity of xi for multinomial data.
transform	the "transform" function that will be called to calculate derivatives, e. g., transform == "saturated" means the function transformSaturated will be called. May be abbreviated.

Details

If differential is missing, the returned value is a new parameter vector of the specified type. If differential is not missing, the returned value is the derivative evaluated at parm and differential, that is, if f is the change-of variable function and ψ is the from parameter, then $f(\psi)$ is calculated when the differential is missing and $f'(\psi)(\delta)$ is calculated when the differential δ is not missing, where the latter is defined by

$$f(\psi + \delta) \approx f(\psi) + f'(\psi)(\delta)$$

for small δ.

The kinds of parameters are "theta" the conditional canonical parameter for the saturated model, "phi" the unconditional canonical parameter for the saturated model, "xi" the conditional mean value parameter for the saturated model, "mu" the unconditional mean value parameter for the saturated model, "beta" the regression coefficient parameter for a canonical affine submodel ($\theta = a + M\beta$ for a conditional canonical affine submodel or $\varphi = a + M\beta$ for an unconditional canonical affine submodel, where a is the offset vector and M is the model matrix), "tau" the mean value parameter for an unconditional canonical affine submodel ($\tau = M^T\mu$, where M is the model matrix).

Only the conditional canonical parameter vector θ and the conditional mean value parameter vector ξ can be checked directly. (To check the validity of another parameter, transform to one of these and check that.) This means that in conversions to these parameters the output vector is checked rather than the input vector, and conversions (apparently) not involving these parameters (which

do go through these parameters inside the transformation function) a conversion to one of these parameters is what is checked rather than the input vector.

There is a difference between conditional and unconditional aster models in the way they treat zero predecessors. For a conditional aster model, if the observed value of the predecessor is zero, then the successor is zero almost surely and can have any parameter value for θ or ξ. For an unconditional aster model, if the expected value of the predecessor is zero, then the successor is zero almost surely and can have any parameter value for θ or ξ.

Since zero values are not allowed at initial nodes (not considered valid by the function validasterdata), the only way predecessor data can be zero almost surely in an unconditional aster model is if the delta vector (`data$redelta`) is not zero so we have a limiting model.

The function `jacobian` turns the derivative considered as a linear transformation calculated by the "transform" functions into the matrix that represents the linear transformation (sometimes called the Jacobian matrix of the transformation). The arguments `modmat` and `offset` are only used if `transform == "conditional"` or `transform == "unconditional"`, and (as with the "transform" functions) the argument `offset` may be missing, in which case the zero vector is used. Not all of the candidate values for `from` and `to` arguments for the `jacobian` function are valid: the value must be valid for the "transform" function that will be called.

Value

a numeric vector of the same length as `parm`. The new parameter if `deriv == FALSE` or the transform of the differential if `deriv = TRUE`. See details.

See Also

asterdata

Examples

```
data(echinacea)
theta <- rnorm(nrow(echinacea$redata), 0, 0.1)
phi <- transformSaturated(theta, echinacea, from = "theta", to = "phi")
## rarely (if ever) want jacobian for unsaturated model transform
## result here is 5130 by 5130 matrix
## Not run: jack <- jacobian(theta, echinacea, from = "theta", to = "phi")
```

Index