

# Package ‘earth’

November 9, 2009

**Version** 2.4-0

**Title** Multivariate Adaptive Regression Spline Models

**Author** Stephen Milborrow derived from mda:mars by Trevor Hastie and Rob Tibshirani.

**Maintainer** Stephen Milborrow <milbo@sonic.net>

**Depends** leaps

**Description** Build regression models using the techniques in Friedman’s papers “Fast MARS” and “Multivariate Adaptive Regression Splines”. (The term “MARS” is copyrighted and thus not used in the name of the package.)

**License** GPL-3

**URL** <http://www.milbo.users.sonic.net>

**Repository** CRAN

**Date/Publication** 2009-11-09 15:40:06

## R topics documented:

contr.earth.response . . . . .	2
earth . . . . .	3
etitanic . . . . .	27
evimp . . . . .	29
format.earth . . . . .	33
mars.to.earth . . . . .	35
model.matrix.earth . . . . .	36
ozone1 . . . . .	38
plot.earth . . . . .	39
plot.earth.models . . . . .	44
plot.evimp . . . . .	46
plotd . . . . .	47
plotmo . . . . .	52
predict.earth . . . . .	57

print.evimp . . . . .	59
residuals.earth . . . . .	60
summary.earth . . . . .	61
update.earth . . . . .	63

<b>Index</b>	<b>65</b>
--------------	-----------

---

contr.earth.response

*Contrasts for the "earth" response*

---

## Description

Contrasts function for factors in the "earth" response. For internal use by earth.

## Usage

```
contr.earth.response(x, base, contrasts)
```

## Arguments

x	a factor
base	unused
contrasts	unused

## Value

Returns a diagonal matrix. An example for a 3 level factor with levels A, B, and C:

```

  A B C
A 1 0 0
B 0 1 0
C 0 0 1
```

## Note

Earth uses this function internally. You shouldn't need it. It is made publicly available only because it seems that is necessary for `model.matrix`.

## See Also

[contrasts](#)

earth

*Multivariate Adaptive Regression Splines***Description**

Build a regression model using the techniques in Friedman's papers "Multivariate Adaptive Regression Splines" and "Fast MARS".

**Usage**

```
## S3 method for class 'formula':
earth(formula = stop("no 'formula' arg"),
      data, weights = NULL, wp = NULL, scale.y = (NCOL(y)==1), subset = NULL,
      na.action = na.fail, glm = NULL, trace = 0,
      keepxy = FALSE, nfold=0, stratify=TRUE, ...)

## Default S3 method:
earth(x = stop("no 'x' arg"), y = stop("no 'y' arg"),
      weights = NULL, wp = NULL, scale.y = (NCOL(y)==1), subset = NULL,
      na.action = na.fail, glm = NULL, trace = 0,
      keepxy = FALSE, nfold=0, stratify=TRUE, ...)

## S3 method for class 'fit':
earth(x = stop("no 'x' arg"), y = stop("no 'y' arg"),
      weights = NULL, wp = NULL, scale.y = (NCOL(y)==1), subset = NULL,
      na.action = na.fail, glm = NULL, trace = 0,
      nk = max(21, 2 * ncol(x) + 1), degree = 1,
      penalty = if(degree > 1) 3 else 2, thresh = 0.001,
      minspan = 0, newvar.penalty = 0, fast.k = 20, fast.beta = 1,
      linpreds = FALSE, allowed = NULL,
      pmethod = "backward", nprune = NULL, Object = NULL,
      Get.crit = get.gcv, Eval.model.subsets = eval.model.subsets,
      Print.pruning.pass = print.pruning.pass, Force.xtx.prune = FALSE,
      Use.beta.cache = TRUE, ...)
```

**Arguments**

To start off, look at the arguments `formula`, `data`, `x`, `y`, `nk`, and `degree`. Many users will find that those arguments are all they need, plus in some cases `keepxy`, `nprune`, `penalty`, `minspan`, and `trace`. For GLM models, use the `glm` argument. For cross validation, use the `nfold` argument.

Model formula.

<code>formula</code>	Data frame for formula.
<code>x</code>	Matrix or dataframe containing the independent variables.
<code>y</code>	Vector containing the response variable, or, in the case of multiple responses, a matrix or dataframe whose columns are the values for each response.

<code>subset</code>	Index vector specifying which cases to use, i.e., which rows in <code>x</code> to use. Default is <code>NULL</code> , meaning all.
<code>weights</code>	Weight vector (not yet supported).
<code>wp</code>	<p>Vector of response weights. Default is <code>NULL</code>, meaning no response weights. If specified, <code>wp</code> must have an element for each column of <code>y</code> (after <code>factors</code>, if any, have been expanded).</p> <p>Note for <code>mda::mars</code> users: <code>earth</code>'s internal normalization of <code>wp</code> is different from <code>mars</code>. <code>Earth</code> uses <code>wp&lt;-sqrt(wp/mean(wp))</code> and <code>mars</code> uses <code>wp&lt;-sqrt(wp/sum(wp))</code>. Thus in <code>earth</code>, a <code>wp</code> with all elements equal is equivalent to no <code>wp</code>. For models built with <code>wp</code>, multiply the GCV calculated by <code>mars</code> by <code>length(wp)</code> to compare it to <code>earth</code>'s GCV.</p>
<code>scale.y</code>	<code>Scale y</code> in the forward pass for better numeric stability. Scaling here means subtract the mean and divide by the standard deviation. Default is <code>NCOL(y)==1</code> , i.e., scale <code>y</code> unless <code>y</code> has multiple columns.
<code>na.action</code>	NA action. Default is <code>na.fail</code> , and only <code>na.fail</code> is supported. (Why? Because adding support to <code>earth</code> for other NA actions is easy, but making sure that is handled correctly internally in <code>predict</code> , <code>plotmo</code> , etc. is tricky. It is more reliable to make the user remove NAs before calling <code>earth</code> .)
<code>glm</code>	<p><code>NULL</code> (default) or a list of arguments to pass on to <code>glm</code>. See the documentation of <code>glm</code> for a description of these arguments (but not all of <code>glm</code>'s arguments are supported, <code>earth</code> will give an error message if you use a <code>glm</code> argument it does not support).</p> <p>See also the "Generalized linear models" section below. Example:  <code>earth(y~x, glm=list(family=binomial))</code></p>
<code>trace</code>	<p>Trace <code>earth</code>'s execution. Default is 0. Values:</p> <p>0 none 0.5 cross validation 1 overview 2 forward pass 3 pruning 4 model mats, memory use, more pruning, etc. 5 and higher for internal details of operation...</p>
<code>keepxy</code>	Set to <code>TRUE</code> to retain the following in the returned value: <code>x</code> and <code>y</code> (or <code>data</code> ), <code>subset</code> , and <code>weights</code> . Default is <code>FALSE</code> . The function <code>update.earth</code> and friends will use these instead of searching for them in the environment at the time <code>update.earth</code> is invoked. This argument also affects the amount of data kept when the <code>nfold</code> argument is used (see <code>cv.list</code> in the "Value" section below).
<b>The following arguments are for the forward pass.</b>	
<code>nk</code>	Maximum number of model terms before pruning, i.e., the maximum number of terms created by the forward pass. Includes the intercept. Default is <code>max(21, 2*NCOL(x)+1)</code> . The number of terms created by the forward pass will be less than <code>nk</code> if a forward stopping condition is reached before <code>nk</code> terms, or if the forward pass drops one side of a hinge pair to prevent linear dependencies. See the "Forward pass" section below.
<code>degree</code>	Maximum degree of interaction (Friedman's <i>mi</i> ). Default is 1, meaning build an additive model (i.e., no interaction terms).
<code>penalty</code>	Generalized Cross Validation (GCV) penalty per knot. Default is <code>if(degree&gt;1) 3 else 2</code> . A value of 0 penalizes only terms, not knots. The value -1 is treated

	<p>specially to mean no penalty, so <math>GCV=RSS/n</math>. Theory suggests values in the range of about 2 to 4. In practice, for big data sets larger values can be useful to force a smaller model. The FAQ section below has some information on GCVs.</p>
thresh	<p>Forward stepping threshold. Default is 0.001. This is one of the arguments used to decide when forward stepping should terminate. See the "Forward pass" section below.</p>
minspan	<p>Minimum distance between knots. (Note: predictor value extremes are ineligible for knots regardless of the minspan setting, as per the MARS paper equation 45.)</p> <p>Use <code>minspan=1</code> to consider all <math>x</math> values (which is good if the data are not noisy).</p> <p>The default is <code>minspan=0</code>. This value 0 is treated specially and means calculate the minspan internally as per Friedman's MARS paper section 3.8 with <math>\alpha = 0.05</math>. Set <code>trace&gt;=2</code> to see the calculated value.</p> <p><b>NEW IN VERSION 2.4-0, Nov 2009:</b> <code>minspan=-1</code> is also treated specially. It is for back compatibility, and means calculate minspan using the (incorrect) method used in versions of earth prior to 2.4-0. Using <code>minspan=-1</code> instead of the default <code>minspan=0</code> will usually build a model with a very similar GCV, although with slightly different knots and terms.</p> <p>The minspan argument is intended to increase resistance to runs of noise in the input data.</p>
newvar.penalty	<p>Penalty for adding a new variable in the forward pass (Friedman's <math>\gamma</math>, equation 74 in the MARS paper). Default is 0, meaning no penalty for adding a new variable. Useful non-zero values range from about 0.01 to 0.2 — you will need to experiment. This argument can mitigate the effects of collinearity or concurvity in the input data, but anecdotal evidence is that it does not work very well. If you know two variables are strongly correlated then you would do better to delete one of them before calling earth.</p>
fast.k	<p>Maximum number of parent terms considered at each step of the forward pass. Friedman invented this parameter to speed up the forward pass (see the Fast MARS paper section 3.0). Default is 20. Values of 0 or less are treated specially (as being equivalent to infinity), meaning no Fast MARS. Typical values, apart from 0, are 20, 10, or 5. In general, with a lower <code>fast.k</code> (say 5), earth is faster; with a higher <code>fast.k</code>, or with <code>fast.k</code> disabled (set to 0), earth builds a better model. However it is not unusual to get a slightly better model with a lower <code>fast.k</code>, and you may need to experiment.</p>
fast.beta	<p>Fast MARS aging coefficient, as described in the Fast MARS paper section 3.1. Default is 1. A value of 0 sometimes gives better results.</p>
linpreds	<p>Index vector specifying which predictors should enter linearly, as in <a href="#">lm</a>.</p> <p>The default is FALSE, meaning all predictors enter in the standard MARS fashion, i.e., in hinge functions.</p> <p>A predictor's index in <code>linpreds</code> is the column number in the input matrix <math>x</math> after factors have been expanded. Examples are given in the FAQ section below. Note: in the current implementation, the GCV penalty for predictors that enter linearly is the same as that for predictors with knots. That is not quite correct; linear terms should be penalized less.</p>

allowed	<p>Function specifying which predictors can interact and how. Default is NULL, meaning all standard MARS terms are allowed.</p> <p>Earth calls the <code>allowed</code> function just before adding a term in the forward pass. If <code>allowed</code> returns TRUE the term goes into the model as usual; if <code>allowed</code> returns FALSE the term is discarded. Examples are given in the FAQ section below.</p> <p>Your <code>allowed</code> function should have the following prototype</p> <pre>allowed &lt;- function(degree, pred, parents, namesx, first) {.....}</pre> <p>where</p> <p><code>degree</code> is the interaction degree of the candidate term. Will be 1 for additive terms.</p> <p><code>pred</code> is the index of the candidate predictor. A predictor's index in <code>linpreds</code> is the column number in the input matrix <code>x</code> after factors have been expanded.</p> <p><code>parents</code> is the candidate parent term's row in <code>dirs</code>.</p> <p><code>namesx</code> is optional and if present is the column names of <code>x</code> after factors have been expanded.</p> <p><code>first</code> is optional and if present is TRUE the first time your <code>allowed</code> function is invoked for the current model, and thereafter FALSE.</p>
nfold	<p>Number of cross validation folds. Default is 0, i.e., no cross validation. If greater than 1, earth first builds a standard model as usual, with all the data. It then builds <code>nfold</code> cross validated models, measuring R-Squared on the left-out data each time. The final cross validation R-Squared is the mean of these R-Squareds. If a binomial or poisson model (see the <code>glm</code> argument), then further statistics are calculated. See the "Cross validation" section below for details.</p>
stratify	<p>Only applies if <code>nfold</code>&gt;1. Default is TRUE. Stratify the cross validation samples so that, for each column of the response <code>y</code> (after factors have been expanded), an approximately equal number of cases with a non-zero response occur in each cross validation subset. That means that if <code>y</code> is a factor, there will be approximately equal numbers of each factor level in each fold (see the "Factors" section below). We say "approximately equal" because the number of occurrences of a factor level may not be exactly divisible by the number of folds.</p>
<b>The following arguments are for the pruning pass.</b>	
pmethod	<p>Pruning method. Default is "backward". One of: <code>backward</code> <code>none</code> <code>exhaustive</code> <code>forward</code> <code>seqrep</code>. Use <code>none</code> to retain all the terms created by the forward pass. If <code>y</code> has multiple columns, then only <code>backward</code> or <code>none</code> is allowed. Pruning can take a while if <code>exhaustive</code> is chosen and the model is big (more than about 30 terms). The current version of the <code>leaps</code> package used during pruning does not allow user interrupts (i.e., you have to kill your R session to interrupt; in Windows hit Ctl-Alt-Delete or from the command line use <code>tskill</code>).</p>
nprune	<p>Maximum number of terms (including intercept) in the pruned model. Default is NULL, meaning all terms created by the forward pass (but typically not all terms will remain after pruning). Use this to reduce exhaustive search time, or to enforce an upper bound on the model size.</p>

**The following arguments are for internal or advanced use.**

Object	Earth object to be updated, for use by <code>update.earth</code> .
Get.crit	Criterion function for model selection during pruning. By default a function that returns the GCV. See the "Pruning pass" section below.
Eval.model.subsets	Function to evaluate model subsets — see notes in source code.
Print.pruning.pass	Function to print pruning pass results — see notes in source code.
Force.xtx.prune	Default is FALSE. This argument pertains to subset evaluation in the pruning pass. By default, if <code>y</code> has a single column then <code>earth</code> calls the <code>leaps</code> routines; if <code>y</code> has multiple columns then <code>earth</code> calls <code>EvalSubsetsUsingXtx</code> . The <code>leaps</code> routines are more accurate but do not support multiple responses ( <code>leaps</code> is based on the QR decomposition and <code>EvalSubsetsUsingXtx</code> is based on the inverse of $X'X$ ). Setting <code>Force.xtx.prune=TRUE</code> forces use of <code>EvalSubsetsUsingXtx</code> , even if <code>y</code> has a single column.
Use.beta.cache	Default is TRUE. Using the "beta cache" takes more memory but is faster (by 20% and often much more for large models). The beta cache uses $nk * nk * ncol(x) * sizeof(double)$ bytes. Set <code>Use.beta.cache=FALSE</code> to save memory. (The beta cache is an innovation in this implementation of MARS and does not appear in Friedman's papers. It is not related to the <code>fast.beta</code> argument.)
...	Dots are passed on to <code>earth.fit</code> .

## Value

An object of class "earth" which is a list with the components listed below. *Term* refers to a term created during the forward pass (each line of the output from `format.earth` is a term, and `dirs` defines which predictors are in which terms). Term number 1 is always the intercept.

<code>rss</code>	Residual sum-of-squares (RSS) of the model (summed over all responses if <code>y</code> has multiple columns).
<code>rsq</code>	$1 - \text{rss} / \text{rss.null}$ . R-Squared of the model (calculated over all responses). A measure of how well the model fits the training data. Note that <code>rss.null</code> is $\text{sum}((y - \text{mean}(y))^2)$ .
<code>gcv</code>	Generalized Cross Validation (GCV) of the model (summed over all responses) The GCV is calculated using the <code>penalty</code> argument. For details of the GCV calculation, see equation 30 in Friedman's MARS paper and <code>earth:::get.gcv</code> .
<code>grsq</code>	$1 - \text{gcv} / \text{gcv.null}$ . An estimate of the predictive power of the model (calculated over all responses). Unlike <code>rsq</code> , in MARS models <code>grsq</code> can be negative. A negative <code>grsq</code> would indicate a severely over parameterized model — a model that would not generalize well even though it may be a good fit to the training data. Watch the GRSq take a nose dive in this example: <code>earth(mpg~., data=mtcars, pmethod="none", trace=3)</code>
<code>bx</code>	Matrix of basis functions applied to <code>x</code> . Each column corresponds to a selected term. Each row corresponds to a row in in the input matrix <code>x</code> , after taking subset. See <code>model.matrix.earth</code> for an example of <code>bx</code> handling. Example <code>bx</code> :

```

      (Intercept) h(Girth-12.9) h(12.9-Girth) h(Girth-12.9)*h(...
[1,]           1           0.0           4.6           0
[2,]           1           0.0           4.3           0
[3,]           1           0.0           4.1           0
...

```

`dirs` Matrix with one row per MARS term, and with with `ij`-th element equal to

0 if predictor `j` is not in term `i`  
 -1 if an expression of the form `pmax(const - xj)` is in term `i`  
 1 if an expression of the form `pmax(xj - const)` is in term `i`  
 2 if predictor `j` enters term `i` linearly.

This matrix includes all terms generated by the `forward.pass`, including those not in `selected.terms`. Note that the terms may not be in pairs, because the forward pass deletes linearly dependent terms before handing control to the pruning pass. Example `dirs`:

```

                                Girth Height
(Intercept)                    0  0 #intercept
h(Girth-12.9)                   1  0 #2nd term uses Girth
h(12.9-Girth)                   -1  0 #3rd term uses Girth
h(Girth-12.9)*h(Height-76)      1  1 #4th term uses Girth and Height
...

```

`cuts` Matrix with `ij`-th element equal to the cut point for predictor `j` in term `i`. This matrix includes all terms generated by the `forward.pass`, including those not in `selected.terms`. Note that the terms may not be in pairs, because the forward pass deletes linearly dependent terms before handing control to the pruning pass. Note for programmers: the precedent is to use `dirs` for term names etc. and to only use `cuts` where cut information needed. Example `cuts`:

```

                                Girth Height
(Intercept)                    0  0 #intercept, no cuts
h(Girth-12.9)                   12.9  0 #2nd term has cut at 12.9
h(12.9-Girth)                   12.9  0 #3rd term has cut at 12.9
h(Girth-12.9)*h(Height-76)      12.9  76 #4th term has two cuts
...

```

`selected.terms` Vector of term numbers in the best model. Can be used as a row index vector into `cuts` and `dirs`. The first element `selected.terms[1]` is always 1, the intercept.

`prune.terms` A matrix specifying which terms appear in which pruning pass subsets. The row index of `prune.terms` is the model size. (The model size is the number of terms in the model. The intercept is counted as a term.) Each row is a vector of term numbers for the best model of that size. An element is 0 if the term is not in the model, thus `prune.terms` is a lower triangular matrix, with dimensions `nprune` x `nprune`. The model selected by the pruning pass is at row number `length(selected.terms)`. Example `prune.terms`:

```

[1,] 1 0 0 0 0 0 0 #intercept-only model
[2,] 1 2 0 0 0 0 0 #best 2 term model uses terms 1,2
[3,] 1 2 4 0 0 0 0 #best 3 term model uses terms 1,2,4
[4,] 1 2 6 9 0 0 0 #and so on
...

```

`rss.per.response` A vector of the RSS for each response. Length is the number of responses, i.e., `ncol(y)` after factors in `y` have been expanded. The `rss` component above is equal to `sum(rss.per.response)`.

`rsq.per.response` A vector of the R-Squared for each response. Length is the number of responses.

`gcv.per.response` A vector of the GCV for each response. Length is the number of responses. The `gcv` component above is equal to `sum(gcv.per.response)`.

`grsq.per.response` A vector of the GRSq for each response. Length is the number of responses.

`rss.per.subset` A vector of the RSS for each model subset generated by the pruning pass. Length is `nprune`. For multiple responses, the RSS is summed over all responses for each subset. The null RSS (i.e., the RSS of an intercept only-model) is `rss.per.subset[1]`. The `rss` above is `rss.per.subset[length(selected.terms)]`.

`gcv.per.subset` A vector of the GCV for each model in `prune.terms`. Length is `nprune`. For multiple responses, the GCV is summed over all responses for each subset. The null GCV (i.e., the GCV of an intercept-only model) is `gcv.per.subset[1]`. The `gcv` above is `gcv.per.subset[length(selected.terms)]`.

`fitted.values` Fitted values. A matrix with dimensions `nrow(y) x ncol(y)` after factors in `y` have been expanded.

`residuals` Residuals. A matrix with dimensions `nrow(y) x ncol(y)` after factors in `y` have been expanded.

`coefficients` Regression coefficients. A matrix with dimensions `length(selected.terms) x ncol(y)` after factors in `y` have been expanded. Each column holds the least squares coefficients from regressing that column of `y` on `bx`. The first row holds the intercept coefficient(s).

`penalty` The GCV penalty used during pruning. A copy of `earth`'s `penalty` argument.

`call` The call used to invoke `earth`.

`terms` Model frame terms. This component exists only if the model was built using `earth.formula`.

`namesx` Column names of `x`, generated internally by `earth` when necessary so each column of `x` has a name. Used, for example, by `predict.earth` to name columns if necessary.

`namesx.org` Original column names of `x`.

levels            Levels of  $y$  if  $y$  is a [factor](#)  
                    $c$  (FALSE, TRUE) if  $y$  is [logical](#)  
                   Else NULL

wp                Copy of the `wp` argument to `earth`.

**The following fields appear only if `earth`'s argument `keepxy` is TRUE.**

`x`

`y`

`data`

`subset`

`weights`        Copies of the corresponding arguments to `earth`. Only exist if `keepxy=TRUE`.

**The following fields appear only if `earth`'s `glm` argument is used.**

`glm.list`        List of GLM models. Each element is the value returned by `earth`'s internal call to [glm](#) for each response.

Thus if there is a single response (or a single binomial pair, see the "Binomial pairs" section below) this will be a one element list and you access the GLM model with `my.earth.model$glm.list[[1]]`.

`glm.coefficients`

GLM regression coefficients. Analogous to the `coefficients` field described above but for the GLM model(s). A matrix with dimensions `length(selected.terms) x ncol(y)` after factors in  $y$  have been expanded. Each column holds the coefficients from the GLM regression of that column of  $y$  on  $bx$ . This duplicates, for convenience, information buried in `glm.list`.

`glm.bpairs`     NULL unless there are paired binomial columns. A logical vector, derived internally by `earth`, or a copy the `bpairs` specified by the user in the `glm` list. See the "Binomial pairs" section below.

**The following fields appear only if the `nfold` argument is greater than 1.**

`cv.rsq.tab`     Matrix with `nfold+1` rows and `nresponse+1` columns, where `nresponse` is the number of responses, i.e., `ncol(y)` after factors in  $y$  have been expanded. The first `nresponse` elements of a row are the RSq's on the left-out data for each response of the model generated at that row's fold. The final column holds the row mean (a weighted mean if `wp` if specified). The final row of the table holds the column means. The values in this final row are the CV-RSqs printed by [summary.earth](#).

Example for a single response model:

```

              y  mean
fold 1  0.909 0.909
fold 2  0.869 0.869
fold 3  0.952 0.952
fold 4  0.157 0.157
fold 5  0.961 0.961
mean    0.769 0.769
```

Example for a multiple response model:

	y1	y2	y3	mean
fold 1	0.915	0.951	0.944	0.937
fold 2	0.962	0.970	0.970	0.968
fold 3	0.914	0.940	0.942	0.932
fold 4	0.907	0.929	0.925	0.920
fold 5	0.947	0.987	0.979	0.971
mean	0.929	0.955	0.952	0.946

`cv.maxerr.tab`

Like `cv.rsq.tab` but is the `MaxErr` at each fold. This is the signed max absolute value at each fold. Also, results are aggregated for the final column and final row using the signed max absolute value instead of the mean. The *signed max absolute value* is defined here as the maximum of the absolute difference between the predicted and observed response values, multiplied by  $-1$  if the sign of the difference is negative.

`cv.deviance.tab`

Like `cv.rsq.tab` but is the `MeanDev` at each fold. Binomial models only.

`cv.calib.int.tab`

Like `cv.rsq.tab` but is the `CalibInt` at each fold. Binomial models only.

`cv.calib.slope.tab`

Like `cv.rsq.tab` but is the `CalibSlope` at each fold. Binomial models only.

`cv.auc.tab`

Like `cv.rsq.tab` but is the `AUC` at each fold. Binomial models only.

`cv.cor.tab`

Like `cv.rsq.tab` but is the `cor` at each fold. Poisson models only.

`cv.nterms`

Vector of length `nfold+1`. Number of MARS terms in the model generated at each cross validation fold, with the final element being the mean of these.

`cv.nvars`

Vector of length `nfold+1`. Number of predictors in the model generated at each cross validation fold, with the final element being the mean of these.

`cv.groups`

Specifies which cases went into which folds. Vector of length equal to the number of cases, with elements taking values in `1:nfold`.

`cv.list`

List of earth models, one model for each fold. These fold models have extra fields `cv.rsq` and `cv.rsq.per.response` (and, if `keepxy` is set, also `cv.test.y` and `cv.test.fitted.values`). To save memory, lengthy fields in the fold models are removed, unless you use the `keepxy` argument. The "lengthy fields" are `$bx`, `$fitted.values`, and `$residuals`.

## Note

Many users will find that it is unnecessary to read this entire section. Just read the parts you need and skim the rest.

## Contents

- . Other implementations
- . Limitations
- . Multiple response models

- . Generalized linear models
- . Factors
- . Binomial pairs
- . The forward pass
- . The pruning pass
- . Execution time
- . Memory use
- . Cross validation
- . Cross validating binomial and poisson models
- . Using earth with fda and mda
- . Migrating from mda::mars
- . Standard model functions
- . Frequently asked questions

### Other implementations

The results are similar to but not identical to other Multivariate Adaptive Regression Splines implementations. The differences stem from the forward pass where very small implementation differences (or perturbations of the input data) can cause rather different selection of terms and knots (although similar GRSq's). The backward passes give identical or near identical results, given the same forward pass results.

The source code of `earth` is derived from `mars` in the `mda` package written by by Trevor Hastie and Robert Tibshirani. See also [mars.to.earth](http://mars.to.earth).

The term "MARS" is trademarked and licensed exclusively to Salford Systems <http://www.salfordsystems.com>. Their implementation uses an engine written by Friedman and has some features not in `earth`.

StatSoft also have an implementation which they call MARSplines <http://www.statsoft.com/textbook/stmars.html>.

### Limitations

The following aspects of MARS are mentioned in Friedman's papers but not implemented in `earth`:

- i) Piecewise cubic models
- ii) Model slicing (`plotmo` goes part way)
- iii) Handling missing values
- iv) Automatic grouping of categorical predictors into subsets
- v) Fast MARS  $h$  parameter

### Multiple response models

If  $y$  has  $k$  columns then `earth` builds  $k$  simultaneous models. (Note that  $y$  will have multiple columns if a factor in  $y$  is expanded by `earth`; see the "Factors" section below for details.) Each model has the same set of basis functions (the same `bx`, `selected.terms`, `dirs` and `cuts`) but different coefficients (the returned `coefficients` will have  $k$  columns). The models are built and pruned as usual but with the GCVs and RSSs averaged across all  $k$  responses.

Since `earth` attempts to optimize for all models simultaneously, the results will not be as "good" as building the models independently, i.e., the GCV of the combined model will usually not be as good as the GCVs for independently built models. However, the combined model may be a better model in other senses, depending on what you are trying to achieve. For example, it could be useful for

earth to select the set of MARS terms that is best across *all* responses. This would typically be the case in a multiple response logistic model if some responses have a very small number of successes.

Note that automatic scaling of  $y$  (via the `scale.y` argument) does not take place if  $y$  has multiple columns. You may want to scale your  $y$  columns before calling `earth` so each  $y$  column gets the appropriate weight during model building (a  $y$  column with a big variance will influence the model more than a column with a small variance). You could do this by calling `scale` before invoking `earth`, or by setting the `scale.y` argument, or by using the `wp` argument.

Here are a couple of (artificial) examples to show some of the ways multiple responses can be specified. Note that `data.frames` can't be used on the left side of a formula, so `cbind` is used in the first example. The examples use the standard technique of specifying a tag `lvol=` to name a column.

```
earth(cbind(Volume, lvol=log(Volume)) ~ ., data=trees)
attach(trees)
earth(data.frame(Girth, Height), data.frame(Volume, lvol=log(Volume)))
```

Don't use a plus sign on the left side of the tilde. You might think that specifies a multiple response, but instead it arithmetically adds the columns.

For more details on using residual errors averaged over multiple responses see section 4.1 of Hastie, Tibshirani, and Buja *Flexible Discriminant Analysis by Optimal Scoring*, JASA, December 1994 <http://www-stat.stanford.edu/~hastie/Papers/fda.pdf>.

### Generalized linear models

Earth builds a GLM model if the `glm` argument is specified. Earth builds the model as usual and then invokes `glm` on the MARS basis matrix `bx`.

In more detail, the model is built as follows. Earth first builds a standard MARS model, including the internal call to `lm.fit` on `bx` after the pruning pass. (See "The forward pass" and "The pruning pass" sections below). Thus knot positions and terms are determined as usual and all the standard fields in `earth`'s return value will be present. Earth then invokes `glm` for the response on `bx` with the parameters specified in the `glm` argument to `earth`. For multiple response models (when  $y$  has multiple columns), the call to `glm` is repeated independently for each response. The results go into three extra fields in `earth`'s return value: `glm.list`, `glm.coefficients`, and `glm.bpairs`.

Earth's internal call to `glm` is made with the `glm` arguments `x`, `y`, and `model` set `TRUE` (see the documentation for `glm` for more information about those arguments).

Use `summary(my.model)` as usual to see the model. Use `summary(my.model, details=TRUE)` to see more details, but note that the printed P-values of the GLM coefficients are meaningless. This is because of the amount of preprocessing by `earth` — the mantra is "variable selection overstates significance of the selected variables". Use `plot(my.model$glm.list[[1]])` to plot the (first) `glm` model.

The examples below show how to specify `earth-glm` models. The examples are only to illustrate the syntax and not necessarily useful models. In the examples `pmethod="none"`, otherwise with these artificial models `earth` tends to prune away everything except the intercept term. You wouldn't normally use `pmethod="none"`. Also, `trace=1`, so if you run these examples you can see how `earth` expands the input matrices (as explained in the "Factors" and "Binomial pairs" sections below).

1. *Two-level factor or logical response.* The response is converted to a single column of 1s and 0s.

```
a1 <- earth(survived ~ ., data=etitanic, # c.f. Harrell "Reg Mod Strat" ch. 12
            degree=2, trace=1,
            glm=list(family=binomial))
```

```
ala <- earth(etitanic[,-2], etitanic[,2], # equivalent but using earth.default
            degree=2, trace=1,
            glm=list(family=binomial))
```

2. *Factor response.* This example is for a factor with more than two levels. (For factors with just two levels, see the previous example.) The factor `pclass` is expanded to three indicator columns (whereas in a direct call to `glm`, `pclass` would be treated as logical: the first level versus all other levels).

```
a2 <- earth(pclass ~ ., data=etitanic, trace=1,
            glm=list(family=binomial))
```

3. *Binomial model* specified with a column pair. This is a single response model but specified with a pair of columns: see the "Binomial pairs" section below. For variety, this example uses a `probit` link and (unnecessarily) increases `maxit`.

```
ldose <- rep(0:5, 2) - 2 # V&R 4th ed. p. 191
sex <- factor(rep(c("male", "female"), times=c(6,6)))
numdead <- c(1,4,9,13,18,20,0,2,6,10,12,16)
pair <- cbind(numdead, numalive=20 - numdead)
a3 <- earth(pair ~ sex + ldose, trace=1, pmethod="none",
            glm=list(family=binomial(link=probit), maxit=100))
```

4. *Double binomial response* (i.e., a multiple response model) specified with two column pairs.

```
numdead2 <- c(2,8,11,12,20,23,0,4,6,16,12,14) # bogus data
doublepair <- cbind(numdead, numalive=20-numdead,
                   numdead2=numdead2, numalive2=30-numdead2)
a4 <- earth(doublepair ~ sex + ldose, trace=1, pmethod="none",
            glm=list(family="binomial"))
```

5. *Poisson model.*

```
counts <- c(18,17,15,20,10,20,25,13,12) # Dobson 1990 p. 93
outcome <- gl(3,1,9)
treatment <- gl(3,3)
a5 <- earth(counts ~ outcome + treatment, trace=1, pmethod="none",
            glm=list(family=poisson))
```

6. *Standard earth model, the long way.*

```
a6 <- earth(numdead ~ sex + ldose, trace=1, pmethod="none",
            glm=list(family=gaussian(link=identity)))
print(a6$coefficients == a6$glm.coefficients) # all TRUE
```

## Factors

**Factors in x:** Earth treats factors in  $x$  in the same way as standard R models such as `lm` (where  $x$  is taken to mean the rhs of the formula). Thus factors are expanded using the current setting of `options("contrasts")`.

**Factors in y:** Earth treats factors in the response in a non-standard way that makes use of earth's ability to handle multiple responses. A *two level factor* (or logical) is converted to a single indicator column of 1s and 0s. A *factor with three or more levels* is converted into  $k$  indicator columns of 1s and 0s, where  $k$  is the number of levels (the `contrasts` matrix is diagonal, see `contr.earth.response`). This happens regardless of the `options("contrasts")` setting and regardless of whether the factors are ordered or unordered. For example, if a column in  $y$  is a factor with levels A, B, and C, the column will be expanded to three columns like this (the actual data will vary but each row will have a single 1):

```
A B C # one column for each factor level
0 1 0 # each row has a single 1
1 0 0
0 0 1
0 0 1
0 0 1
...
```

In distinction, a standard `treatment contrast` on the rhs of a model with an intercept would have no first "A" column (to prevent linear dependencies on the rhs of the model formula).

This expansion to multiple columns (which only happens for factors with more than two levels) means that earth will build a multiple response model as described in the "Multiple responses" section above.

Paired binomial response columns in  $y$  are treated specially — see the "Binomial pairs" section below.

Use `trace=1` or higher to see the column names of the  $x$  and  $y$  matrices after factor processing. Use `trace=4` to see the first few rows of  $x$  and  $y$  after factor processing.

Here is an example which uses the `etitanic` data to predict the passenger class (not necessarily a sensible thing to do but provides a good example here):

```
> data(etitanic)
> head(etitanic) # pclass and sex are unordered factors

  pclass survived    sex    age sibsp parch
1     1st         1 female 29.000     0     0
2     1st         1  male  0.917     1     2
3     1st         0 female  2.000     1     2

> earth(pclass ~ ., data=etitanic, trace=1) # note col names in x and y below

x is a 1046 by 5 matrix: 1=survived, 2=sexmale, 3=age, 4=sibsp, 5=parch
y is a 1046 by 3 matrix: 1=1st, 2=2nd, 3=3rd
rest not shown here...
```

### Binomial pairs

This section is only relevant if you use earth's `glm` argument with a binomial or quasibinomial family.

Users of the `glm` function will be familiar with the technique of specifying a binomial response as a two-column matrix, with a column for the number of successes and a column for the failures. Earth automatically detects when such columns are present in `y` (by looking for adjacent columns which both have entries greater than 1). The first column only is used to build the standard earth model. Both columns are then passed to earth's internal call to `glm`. As always, use `trace=1` to see how the columns of `x` and `y` are expanded.

You can override this automatic detection by including a `bpairs` parameter. This is usually (always?) unnecessary. For example

```
glm=list(family=binomial, bpairs=c(TRUE, FALSE))
```

specifies that there are two columns in the response with the second paired with the first. These examples

```
glm=list(family=binomial, bpairs=c(TRUE, FALSE, TRUE, FALSE))
glm=list(family=binomial, bpairs=c(1,3)) # equivalent
```

specify that the 1st and 2nd columns are a binomial pair and the 3rd and 4th columns another binomial pair.

### The forward pass

Understanding the details of the forward and pruning passes will help you understand earth's return value and the admittedly large number of arguments. The result of the forward pass is the MARS basis matrix `bx` and the set of terms defined by `dirs` and `cuts` (these are all fields in earth's return value, but the `bx` here includes all terms before trimming back to `selected.terms`).

The forward pass adds terms in pairs until the first of the following conditions is met:

- i) reach maximum number of terms (`nterms >= nk`)
- ii) reach DeltaRSq threshold (`DeltaRSq < thresh`), where DeltaRSq is the difference in R-Squared caused by adding the current term pair, and `thresh` is the argument to earth
- iii) reach max RSq (`RSq > 1-thresh`)
- iv) reach min GRSq (`GRSq < -10`) (-10 is a pathologically bad GRSq)
- v) no new term increases the RSq (reached numerical limits).

Set `trace>=1` to see the stopping condition and `trace>=2` to trace the forward pass.

You can disable all termination conditions except (i) and (v) by setting `thresh=0`. See the FAQ below "Why do I get fewer terms than `nk`?"

Note that GCVs (via GRSq) are used during the forward pass only as one of the (more unusual) stopping conditions and in `trace` prints. Changing the `penalty` argument does not change the knot positions.

The various stopping conditions mean that the actual number of terms created by the forward pass may be less than `nk`. There are other reasons why the actual number of terms may be less than `nk`: (i) the forward pass discards one side of a term pair if it adds nothing to the model — but the forward pass counts terms as if they were actually created in pairs, and, (ii) as a final step, the forward pass deletes linearly dependent terms, if any, so all terms in `dirs` and `cuts` are independent. And remember that the pruning pass will further discard terms.

### The pruning pass

The pruning pass is handed the sets of terms created by the forward pass. Its job is to find the subset of those terms that gives the lowest GCV. The following description of the pruning pass explains how various fields in earth's returned value are generated.

The pruning pass works like this: it determines the subset of terms in `bx` (using `pmethod`) with the lowest RSS for each model size in `1:nprune` (see the `Force.xtx.prune` argument above for some details). It saves the RSS and term numbers for each such subset in `rss.per.subset` and `prune.terms`. It then applies the `Get.crit` function with `penalty` to `rss.per.subset` to yield `gcv.per.subset`. Finally it chooses the model with the lowest value in `gcv.per.subset`, puts its term numbers into `selected.terms`, and updates `bx` by keeping only the `selected.terms`.

After the pruning pass, earth runs `lm.fit` to determine the `fitted.values`, residuals, and coefficients, by regressing the response `y` on `bx`. If `y` has multiple columns then `lm.fit` is called for each column.

If a `glm` argument is passed to earth, earth runs `glm` on (each column of) `y` in addition to the above call to `lm.fit`.

Set `trace>=3` to trace the pruning pass.

By default `Get.crit` is `earth:::get.gcv`. Alternative `Get.crit` functions can be defined. See the source code of `get.gcv` for an example.

### Execution time: "I wanna go fast"

For a given set of input data, the following can increase the speed of the forward pass:

- i) decreasing `fast.k`
- ii) decreasing `nk` (because fewer forward pass terms)
- iii) decreasing `degree`
- iv) increasing `thresh` (because fewer forward pass terms)
- v) increasing `min.span`.

The backward pass is normally much faster than the forward pass, unless `pmethod="exhaustive"`. Reducing `nprune` reduces exhaustive search time. One strategy is to first build a large model and then adjust pruning parameters such as `nprune` using `update.earth`.

The following very rough rules of thumb apply for large models. Using `minspan=1` instead of the default 0 will increase times by 20 to 50%. Using `fast.k=5` instead of the default 20 can give substantial speed gains but will sometimes give a much smaller GRSq. Using an `allowed` function slows down model building by about 10%.

### Memory use

Earth does not impose specific limits on the model size. Model size is limited only by the amount of memory on your system, the maximum memory addressable by R, and your patience. On a 32 bit machine with `x` and `y` of type double (no factors), the number of bytes of memory used by earth is about

$$8 * (nk^2 * ncol(x) + (nrow(x) * (3 + 2*nk + ncol(x)/2))).$$

Earth prints the results of the above calculation if `trace>=4`. Memory use peaks in the forward pass. The bulk of the forward pass is implemented in C. It allocates memory "outside of R" and so `memory.size` will not report the memory it uses.

Before calling `earth`, R itself will of course allocate memory over and above the amount calculated above. To reduce total memory usage, it sometimes helps to `remove` variables and call `gc` before invoking `earth`.

Earth uses more memory if any elements of the `x` and `y` arguments are not `double`, because it must convert them to double internally. The same applies if the `subset` argument is used. Earth uses more memory if `trace`  $\geq 2$  (because `DUP=TRUE` is required to pass predictor names to earth's internal call to `.C`). Increasing the `degree` does not change the memory requirement but greatly increases the running time.

Here is an example of memory use: the earth test suite builds a model using `earth.default` with a  $1e4$  by 100 input matrix with `nk=21`. The Windows XP task manager reports that the peak memory use when building this model is 47 MBytes. Using the formula interface to earth pushes memory to 62 MBytes. Increasing the number of rows in the input matrix to  $1e5$  pushes memory to 240 MBytes.

### Cross validation

Use cross validation to get an estimate of RSq on independent data. Example (note the `nfold` parameter):

```
a <- earth(survived ~ ., data=etitanic, degree=2, nfold=10)
summary(a) # note the CV-RSq field
```

Cross validation is done if `nfold` is greater than 1 (typically 10). Earth first builds a standard model with all the data as usual. This means that all the standard fields in earth's return value appear as usual. Earth then builds `nfold` cross validated models. It measures RSq on the test data (i.e., the left-out data) for each fold. The final cross validation RSq is the mean of these RSq's. Use `summary.earth` to see this final value and its standard deviation across cross validation folds.

The cross validation results go into extra fields in earth's return value. All of these have a `cv` prefix — see the "Value" section above for details.

For multiple response models, at each fold earth calculates the RSq for each response independently, and combines these by taking their mean (or weighted mean if the `wp` argument is used).

With `trace=.5` or higher, earth prints out progress information as cross validation proceeds. For example

```
CV fold 3: CV-RSq 0.622  ntrain-nz 384  ntest-nz 43
```

shows that for cross validation fold number 3, the RSq on the test set (i.e., the left-out data) is 0.622. The printout also shows the number of non-zero values in the observed response in the fold's training set and test set. This is useful if you have a binary or factor response and want to check that you have enough examples of each factor level in each fold. With the `stratify` argument (which is enabled by default), earth attempts to keep the numbers of each level constant across folds.

For reproducibility, call `set.seed` before calling earth with `nfold`.

### Cross validating binomial and poisson models

If you cross validate a binomial or poisson model (specified using earth's `nfold` and `glm` arguments), earth returns the following additional statistics. Each of these is measured on the test set for each fold, and averaged across all folds (except that the signed max absolute value instead of the average is used for `MaxErr`). Use `summary.earth` to see these statistics and their standard deviation across folds.

**CV-RSq** cross validated R-Squared, identical to CV-RSq for non-glm models

**MaxErr** signed max absolute difference between the predicted and observed response. This is the maximum of the absolute differences between the predicted and observed response values, multiplied by  $-1$  if the sign of the difference is negative.

**MeanDev** deviation divided by the response length

**CalibInt CalibSlope** calibration intercept and slope (from regressing the observed response on the predicted response)

**AUC** (binomial models only) area under the ROC curve

**cor** (poisson models only) correlation between the predicted and observed response

See the source code in `earth.cv.R` for details.

For multiple response models, at each fold earth calculates these statistics for each response independently, and combines them by taking their mean, or weighted mean if the `wp` argument is used (but takes the signed max absolute value instead of the mean for `MaxErr`) [TODO should do the same for `CalibInt`, `CalibSlope`?]. Taking the mean is a rather dubious way of combining results from what are essentially quite different models, but can nevertheless be useful.

Explanations of the above statistics can be found in the following (and many other) references:

T. Fawcett (2004) *ROC Graphs: Notes and Practical Considerations for Researchers*. Revised version of Technical report HP Laboratories. [http://home.comcast.net/~tom.fawcett/public\\_html/papers](http://home.comcast.net/~tom.fawcett/public_html/papers)

J. Pearce and S. Ferrier (2000) *Evaluating the predictive performance of habitat models developed using logistic regression*

F. Harrell (2001) *Regression Modeling Strategies with Applications to Linear Models, Logistic Regression, and Survival Analysis* <http://biostat.mc.vanderbilt.edu/twiki/bin/view/Main/RmS>

### Using earth with fda and mda

Earth can be used with `fda` and `mda` in the `mda` package. Earth will generate a multiple response model. Use the `fda/mda` argument `keep.fitted=TRUE` if you want to call `plot.earth` later (actually only necessary for large datasets, see the description of `keep.fitted` in `fda`). Use the `earth` argument `keepxy=TRUE` if you want to call `update.earth` or `plotmo` later. Example:

```
library(mda)
(a <- fda(Species~., data=iris, keep.fitted=TRUE, method=earth, keepxy=TRUE))
plot(a)
summary(a$fit) # examine earth model embedded in fda model
plot(a$fit)
plotmo(a$fit, ycolumn=1, ylim=c(-1.5,1.5), clip=FALSE)
plotmo(a$fit, ycolumn=2, ylim=c(-1.5,1.5), clip=FALSE)
```

### Migrating from mda::mars

Changing code from `mda::mars` to `earth` is usually just a matter of changing the call from "mars" to "earth". But there are a few argument differences and earth will issue a warning if you give it a mars-only argument.

The resulting model will be similar but not identical because of small implementation differences which are magnified by the inherent instability of the MARS forward pass.

If you are further processing the output of `earth` you will need to consider differences in the returned value. The header of the source file `mars.to.earth.R` describes these. Perhaps the most important is that `mars` returns the MARS basis matrix in a field called "x" whereas `earth` returns "bx". Also, `earth` returns "dirs" rather than "factors", and in `earth` this matrix can have entries of value 2 for linear predictors.

See also [mars.to.earth](#).

### Standard model functions

Standard model functions such as [case.names](#) are provided for `earth` objects and are not explicitly documented. Many of these give warnings when the results are not what you may expect. Pass `warn=FALSE` to these functions to turn off just these warnings.

## FREQUENTLY ASKED QUESTIONS

### What are your plans for earth?

We would like to add support of case weights (to allow boosting), but that won't happen anytime soon.

### How can I establish variable importance?

Use the [evimp](#) function. See its help page for more details.

The [summary.earth](#) function lists the predictors in order of estimated importance using the `nsubsets` criterion of [evimp](#).

### Which predictors were added to the model first?

You can see the forward pass adding terms with `trace=2` or higher. But remember, pruning will usually remove some of the terms. You can also use

```
summary(my.model, decomp="none")
```

which will list the basis functions remaining after pruning, in the order they were added by the forward pass.

### Which predictors are actually in the model?

The following function will give a list of predictors in the model:

```
get.used.pred.names <- function(obj) # obj is an earth object
{
  any1 <- function(x) any(x != 0) # like any but no warning if x is double
  names(which(apply(obj$dirs[obj$selected.terms,,drop=FALSE],2,any1)))
}
```

### How can I train on one set of data and test on another?

The example below demonstrates one way to train on 80% of the data and test on the remaining 20%.

```

train.subset <- sample(1:nrow(trees), .8 * nrow(trees))
test.subset <- (1:nrow(trees))[-train.subset]
a <- earth(Volume ~ ., data = trees[train.subset, ])
yhat <- predict(a, newdata = trees[test.subset, ])
y <- trees$Volume[test.subset]
print(1 - sum((y - yhat)^2) / sum((y - mean(y))^2)) # print R-Squared

```

In practice a dataset larger than the one in the example should be used for splitting. The model variance is too high with this small set — run the example a few times to see how the model changes as `sample` splits the dataset differently on each run. Also, remember that the test set should not be used for parameter tuning because you will be optimizing for the test set — instead use GCVs, separate parameter selection sets, or techniques such as cross-validation.

#### Why do I get fewer terms than `nk`, even with `prune="none"`?

There are several conditions that can terminate the forward pass, and reaching `nk` is just one of them. See the "Forward pass" section above.

Setting `earth`'s argument `thresh` to zero is treated as a special case: `thresh=0` disables all termination conditions except `nk` and conditions involving numerical limits. With `thresh=0`, the measured GRSq (and thus the efficacy of the pruning pass) should be treated with skepticism, especially if you get the warning

```
effective number of GCV parameters >= number of cases.
```

#### Why do I get fewer terms than `nprune`?

The pruning pass selects a model with the lowest GCV that has `nprune` or fewer terms. Thus the `nprune` argument specifies the *maximum* number of permissible terms in the final pruned model.

You can work around this because you will get exactly `nprune` terms if you specify `pmethod="none"`. Compare the output of these two examples:

```

earth(Volume ~ ., data = trees, trace=3)
earth(Volume ~ ., data = trees, trace=3, pmethod="none")

```

Another way to get exactly `nprune` terms is to specify `penalty = -1`. This special value of `penalty` causes `earth` to set the GCV to  $RSS/nrow(x)$ . Since the training RSS always decreases with more terms, the pruning pass will choose the maximum allowable number of terms. An example:

```
earth(Volume ~ ., data = trees, trace=3, penalty=-1)
```

#### Is it best to hold down model size with `nk` or `nprune`?

If you want the best possible small model, build a big model (by specifying a big `nk`) and prune it back (by specifying a small `nprune`). This is better than directly building a small model by specifying a small `nk`, because the pruning pass can look at all the terms whereas the forward pass can only see one term ahead. However, it is much faster building a small model by specifying a small `nk`.

#### Can you give an example of the `linpreds` argument?

With `linpreds` you can specify which predictors should enter linearly, instead of in hinge functions. The `linpreds` argument does not stipulate that a predictor *must* enter the model, only that if it enters it should enter linearly. Starting with

```
a1 <- earth(Volume ~ ., data = trees)
plotmo(a1)
```

we see in the `plotmo` graphs or by running `evimp` that Height is not as important as Girth. For collaborative evidence that Girth is a more reliable indicator of Volume you can use `pairs`:

```
pairs(trees, panel = panel.smooth)
```

Since we want the simplest model that describes the data, we can specify that Height should enter linearly:

```
a2 <- earth(Volume ~ ., data = trees, linpreds = 2) # 2 is Height column
summary(a2)
```

which yields

```

                coefficients
(Intercept)      2.981
Height           0.348
h(Girth-14)      6.302
h(14-Girth)     -3.128
```

In this example, the second simpler model has almost the same RSS as the first model. We can make both Girth and Height enter linearly with

```
a3 <- earth(Volume ~ ., data = trees, linpreds = c(1,2))
```

or with (the single TRUE is recycled to the length of linpreds)

```
a4 <- earth(Volume ~ ., data = trees, linpreds = TRUE)
```

But specifying that all predictors should enter linearly is not really a useful thing to do. In our simple example, the all-linear MARS model is the same as a standard linear model

```
a5 <- lm(Volume ~ ., data = trees)
```

(compare the `summary` for each) but in general that will not be true. Earth will not include a linear predictor if that predictor does not improve the model.

### Can you give an example of the `allowed` argument?

You can specify how variables are allowed to enter MARS terms with the `allowed` argument.

The interface is flexible but requires a bit of programming. We start with a simple example, which completely excludes one predictor from the model:

```
example1 <- function(degree, pred, parents) # returns TRUE if allowed
{
  pred != 2 # disallow predictor 2, which is "Height"
}
a1 <- earth(Volume ~ ., data = trees, allowed = example1)
print(summary(a1))
```

But that's not much use, because it's simpler to exclude the predictor from the input matrix when invoking `earth`:

```
a2 <- earth(Volume ~ . - Height, data = trees)
```

The example below is more useful. It prevents the specified predictor from being used in interaction terms. (The example is artificial because it's unlikely you would want to single out humidity from interactions in the ozone data.)

The `parents` argument is the candidate parent's row in the `dirs` matrix (`dirs` is described in the "Value" section above). Each entry of `parents` is 0, 1, -1, or 2, and you index `parents` on the predictor index. Thus `parents[pred]` is 0 if `pred` is not in the parent term.

```
example2 <- function(degree, pred, parents)
{
  # disallow humidity in terms of degree > 1
  # 3 is the "humidity" column in the input matrix
  if (degree > 1 && (pred == 3 || parents[3]))
    return(FALSE)
  TRUE
}
a3 <- earth(O3 ~ ., data = ozonel, degree = 2, allowed = example2)
print(summary(a3))
```

The following example allows only the specified predictors in interaction terms:

```
example3 <- function(degree, pred, parents)
{
  # allow only humidity and temp in terms of degree > 1
  # 3 and 4 are the "humidity" and "temp" columns
  allowed.set = c(3,4)
  if (degree > 1 &&
      (all(pred != allowed.set) || any(parents[-allowed.set])))
    return(FALSE)
  TRUE
}
a4 <- earth(O3 ~ ., data = ozonel, degree = 2, allowed = example3)
print(summary(a4))
```

**Further notes.** The basic MARS model building strategy is always applied even when there is an `allowed` function. For example, `earth` considers a term for addition only if all factors of that term except the new one are already in a model term. This means that an `allowed` function that inhibits, say, all degree 2 terms will also effectively inhibit higher degrees too, because there will be no degree 2 terms for `earth` to extend to degree 3.

You can expect model building to be about 10% slower with an `allowed` function because of the time taken to invoke the `allowed` function. On the other hand, execution time can sometimes be faster because you are evaluating fewer potential MARS terms.

**Using predictor names instead of indices in the "allowed" function.** You can use predictor names instead of indices using the optional `namesx` argument. If present, `namesx` is the column names of `x` after factors have been expanded. The first example above (the one that disallows `Height`) can be rewritten as

```
example1a <- function(degree, pred, parents, namesx)
{
  namesx[pred] != "Height"
}
```

Comparing strings is inefficient and the above example can be rewritten a little more efficiently using the optional `first` argument. If present, this is TRUE the first time your allowed function is called for the current model and thereafter FALSE.

```
iheight <- 0 # column index of "Height"

example1b <- function(degree, pred, parents, namesx, first)
{
  if (first) {
    # first time this function is invoked, so
    # stash column index of "Height" in iheight
    iheight <<- which(namesx == "Height") # note use of <<- not <-
    if (length(iheight) != 1) # sanity check
      stop("no Height in ", paste(namesx, collapse=" "))
  }
  pred != iheight
}
```

### How does `summary.earth` order terms?

With `decomp="none"`, the terms are ordered as created by the forward pass.

With the default `decomp="anova"`, the terms are ordered in increasing order of interaction. In detail:

- (i) terms are sorted first on degree of interaction
- (ii) then terms with a `linpreds` linear factor before standard terms
- (iii) then on the predictors (in the order of the columns in the input matrix)
- (ii) and finally on increasing knot values.

It's actually `earth:::reorder.earth` that does the ordering.

`summary.earth` lists predictors with weird names that aren't in `x`. What gives?

You probably have factors in your `x` matrix, and `earth` is applying `contrasts`. See the "Factors" section above.

### Why `pmax` and not `max` in the output from `summary.earth` (with `style="pmax"`)?

With `pmax` the `earth` equation is an R expression that can handle multiple cases. Thus the expression is consistent with the way `predict.earth` works — you can give `predict` multiple cases (i.e., multiple rows in the input matrix) and it will return a vector of predicted values.

### What about boosting MARS?

If you want to boost, use boosted trees rather than boosted MARS — you will get better results.

More precisely, although gradient boosted MARS gives better results than plain MARS, if you would like to improve prediction performance (at the cost of a more complicated and less interpretable model) you will usually get better results with boosted trees (via, say, the `gbm` package)

than with boosted MARS. See Gillian Ward (2007) *Statistics in Ecological Modeling: Presence-Only Data and Boosted Mars (Doctoral Thesis)* [http://www-stat.stanford.edu/~hastie/THESES/Gill\\_Ward.pdf](http://www-stat.stanford.edu/~hastie/THESES/Gill_Ward.pdf).

This could change as the state of the art advances.

### What about bagging MARS?

The `caret` package provides functions for bagging earth (and for parameter selection). Our personal experience has been that bagging earth does not give models with better predictive ability. Your mileage may vary. (We used a modified version of earth that randomized the set of variables available at each forward step, as well as trying other approaches).

### What is a GCV, in simple terms?

GCVs are important for MARS because the pruning pass uses GCVs to evaluate model subsets.

In general terms, when testing a model (not necessarily a MARS model) we want to test *generalization* performance and so want to measure error on independent data, i.e., not on the training data. Often a decent set of independent data is unavailable and so we resort to cross validation or leave-one-out methods. But that can be painfully slow. As an alternative, for certain forms of model we can use a formula to approximate the error that would be determined by leave-one-out validation — that approximation is the GCV. The formula adjusts (i.e., increases) the training RSS to take into account the flexibility of the model. Summarizing, the GCV approximates the RSS (divided by the number of cases) that would be measured on independent data. Even when the approximation is not that good, it is usually good enough for comparing models during pruning.

GCVs were introduced by Craven and Wahba, and extended by Friedman for MARS. See Hastie et al. p216 and the Friedman MARS paper. GCV stands for "Generalized Cross Validation", a perhaps misleading term.

The `GRSq` measure used in the earth package standardizes the raw GCV, in the same way that R-Squared standardizes the RSS.

### If GCVs are so important, why don't linear models use them?

First a few words about overfitting. An overfit model fits the training data well but will not give good predictions on new data. The idea is that the training data captures the underlying structure in the system being modeled, plus noise. We want to model the underlying structure and ignore the noise. An overfit model models the specific realization of noise in the training data and thus is too specific to the training data.

The more flexible a model, the more its propensity to overfit the training data. Linear models are constrained, with usually only a few parameters, and don't have the tendency to overfit like more flexible models such as MARS. This means that for linear models, the RSS on the data used to build the model is usually an adequate measure of generalization ability.

This is no longer true if you do automatic variable selection on linear models, because the process of selecting variables increases the flexibility of the model. Hence the AIC — as used in, say, `drop1`. The GCV, AIC, and friends are means to the same end. Depending on what information is available during model building, we use one of these statistics to estimate model generalization performance for the purpose of selecting a model.

### What happened to `get.nterms.per.degree`, `get.nused.preds.per.subset`, and `reorder.earth`?

From release 1.3.0, some earth functions are no longer public, to help simplify the user interface.

The functions are still available (and stable) if you need them — use for example `earth:::reorder.earth()`.

### What happened to the `ppenalty` argument?

This was removed (release 1.3.1) because it is no longer needed. The only argument of `update.earth` is a more flexible way of achieving the same end.

### Author(s)

Stephen Milborrow, derived from `mda::mars` by Trevor Hastie and Robert Tibshirani.

The approach used for GLMs was motivated by work done by Jane Elith and John Leathwick (a representative paper is listed in the references below).

The `evimp` function uses ideas from Max Kuhn's `caret` package <http://cran.r-project.org/web/packages/caret/index.html>.

Users are encouraged to send feedback — use milbo AT sonic PERIOD net <http://www.milbo.users.sonic.net>.

### References

The primary references are the Friedman papers. Readers may find the MARS section in Hastie, Tibshirani, and Friedman a more accessible introduction. The Wikipedia article is recommended for an elementary introduction. Faraway takes a hands-on approach, using the `ozone` data to compare `mda::mars` with other techniques. (If you use Faraway's examples with `earth` instead of `mars`, use `$bx` instead of `$x`.) Friedman and Silverman is recommended background reading for the MARS paper. Earth's pruning pass uses the `leaps` package which is based on techniques in Miller.

Faraway (2005) *Extending the Linear Model with R* <http://www.maths.bath.ac.uk/~jjf23>

Friedman (1991) *Multivariate Adaptive Regression Splines (with discussion)* Annals of Statistics 19/1, 1–141 <http://www.salfordsystems.com/doc/MARS.pdf>

Friedman (1993) *Fast MARS* Stanford University Department of Statistics, Technical Report 110 <http://www.milbo.users.sonic.net/earth/Friedman-FastMars.pdf>, <http://www-stat.stanford.edu/research/index.html>

Friedman and Silverman (1989) *Flexible Parsimonious Smoothing and Additive Modeling* Technometrics, Vol. 31, No. 1. <http://links.jstor.org/sici?sici=0040-1706%28198902%2931%3A1%3C3%3AFPSAAM%3E2.0.CO%3B2-Z>

Hastie, Tibshirani, and Friedman (2001) *The Elements of Statistical Learning* <http://www-stat.stanford.edu/~hastie/pub.htm>

Leathwick, J.R., Rowe, D., Richardson, J., Elith, J., & Hastie, T. (2005) *Using multivariate adaptive regression splines to predict the distributions of New Zealand's freshwater diadromous fish* Freshwater Biology, 50, 2034-2052 <http://www-stat.stanford.edu/~hastie/pub.htm>, <http://www.botany.unimelb.edu.au/envisci/about/staff/elith.html>

Miller, Alan (1990, 2nd ed. 2002) *Subset Selection in Regression* <http://users.bigpond.net.au/amiller>

Wikipedia article on MARS [http://en.wikipedia.org/wiki/Multivariate\\_adaptive\\_regression\\_splines](http://en.wikipedia.org/wiki/Multivariate_adaptive_regression_splines)

**See Also**

Start with [summary.earth](#), [plot.earth](#), [plotmo](#), and [evimp](#).

```
etitanic
evimp
format.earth
mars.to.earth
model.matrix.earth
ozonel
plot.earth.models
plot.earth
plotd
plotmo
predict.earth
residuals.earth
summary.earth
update.earth
```

**Examples**

```
a <- earth(Volume ~ ., data = trees)
summary(a, digits = 2, style = "pmax")

# yields:
# Call: earth(formula=Volume~., data=trees)
#
# Volume =
# 27
# + 6.2 * pmax(0, Girth - 14)
# - 3.3 * pmax(0, 14 - Girth)
# + 0.49 * pmax(0, Height - 72)
#
# Selected 4 of 6 terms, and 2 of 2 predictors
# Importance: Girth, Height
# Number of terms at each degree of interaction: 1 3 (additive model)
# GCV 11    RSS 197    GRSq 0.96    RSq 0.98
```

---

etitanic

*Titanic data with incomplete cases removed*

---

**Description**

Titanic data with incomplete cases, passenger names, and other details removed.

**Format**

A data frame with 1046 observations on 6 variables.

pclass      passenger class, unordered factor: 1st 2nd 3rd

```

survived integer: 0 or 1
sex       unordered factor: male female
age       age in years, double: min 0.167 max 80.0
sibsp    number of siblings or spouses aboard, integer: 0..8
parch    number of parents or children aboard, integer: 0..6

```

## Source

This dataset is included in the earth package because it is a convenient vehicle for illustrating earth's GLM and factor handling.

The dataset was compiled by Frank Harrell and Robert Dawson: <http://biostat.mc.vanderbilt.edu/twiki/pub/Main/DataSets/titanic.html>

See also:

<http://biostat.mc.vanderbilt.edu/twiki/pub/Main/DataSets/titanic3info.txt>.

For this version of the Titanic data, passenger details and incomplete cases were deleted and the name changed to `etitanic` to minimize confusion with other versions ("e" because it is part of the earth package).

Contents of `etitanic`:

```

      pclass survived      sex      age sibsp parch
1         1st         1 female 29.000     0     0
2         1st         1  male  0.917     1     2
3         1st         0 female  2.000     1     2
4         1st         0  male 30.000     1     2
5         1st         0 female 25.000     1     2
...
1309      3rd         0  male 29.000     0     0

```

How `etitanic` was built:

```

load("titanic3") # from Harrell's web site
# discard name, ticket, fare, cabin, embarked, body, home.dest
etitanic = titanic3[,c(1,2,4,5,6,7)]
etitanic = etitanic[!is.na(etitanic$age),]
save(etitanic, file="etitanic.rda")

```

## See Also

[earth](#)

---

 evimp

*Estimate variable importances in an "earth" object*


---

## Description

Estimate variable importances in an `earth` object

## Usage

```
evimp(obj, trim=TRUE, sqrt.=FALSE)
```

## Arguments

<code>obj</code>	An <code>earth</code> object.
<code>trim</code>	If TRUE (default), delete rows in the returned matrix for variables that don't appear in any subsets.
<code>sqrt.</code>	Default is FALSE. If TRUE, take the <code>sqrt</code> of the GCV and RSS importances before normalizing to 0 to 100. This arguably gives a better indication of relative importances because the raw importances are calculated using a sum of squares.

## Value

See the example later. This function returns matrix showing the relative importances of the variables in the model. There is a row for each variable. The row name is the variable name, but with `-unused` appended if the variable does not appear in the final model.

The columns of the matrix are:

`col`: column index of the variable in the `x` argument to `earth`.

`used`: 1 if the variable is used in the final model, else 0. Equivalently, 0 if the row name has a `-unused` suffix.

`nsubsets`: variable importance using the "number of subsets" criterion. Is the number of subsets that include the variable (see "Three Criteria" below).

`gcv`: variable importance using the GCV criterion (see below).

`rss`: variable importance using the RSS criterion (see below).

The rows are sorted on the `nsubsets` criterion. This means that values in the `nsubsets` column decrease as you go down the column (more accurately, they are non-increasing). The values in the `gcv` and `rss` columns are also non-increasing, except where the `gcv` or `rss` ranking differs from the `nsubsets` ranking.

For convenience scanning the columns by eye, there are unnamed columns (not listed above) after the `gcv` column and the `rss` column. These have a 0 where the ranking using the `gcv` or `rss` criteria differs from that using the `nsubsets` criterion. In other words, there is a 0 for values that increase as you go down the `gcv` or `rss` column.

**Note****Introduction to variable importance**

What exactly is variable importance? A working definition is that a variable's importance is a measure of the effect that observed changes to the variable have on the observed response. It is this measure of importance that `evimp` tries to estimate.

Variable importance in the *equation that MARS derives from the data* is not quite the same thing. For example, if two variables are highly correlated, MARS will usually drop one when building the model. Both variables have the same importance in the data but not in the MARS equation (one variable does not even appear in the equation). A section below has a few words on how to use `plotmo` to estimate variable importance in the MARS equation.

You might say that you can measure a variable's importance by changing the variable's value and measuring how the response changes. However, except in special situations, there are problems with this because:

- (i) it assumes we can change the variable, which is usually not the case. For example, in the `trees` data, we cannot simply generate a new tree of arbitrary height.
- (ii) it assumes that changes to a variable occur in isolation. In practice, a variable is usually tied to other variables, and a change to the variable would never occur without simultaneous changes to other variables. For example, in the `trees` data, a change to the height is associated with a change in the girth.

[Note: this section was written in response to several emails about `evimp`.

Your comments would be appreciated.]

**Estimating variable importance**

Establishing predictor importance is in general a tricky and even controversial problem. There is no completely reliable way to estimate the importance of the variables in a standard MARS model, unless you make further lengthy tests after the model is built (lengthy tests such as leave-one-out techniques, see the section below on building many models). The `evimp` function just makes an educated (and in practice useful) guess as described below.

**Three criteria for estimating variable importance**

The `evimp` functions uses three criteria for estimating variable importance.

1. The `nsubsets` criterion counts the number of model subsets that include the variable. Variables that are included in more subsets are considered more important.

By "subsets" we mean the subsets of terms generated by the pruning pass. There is one subset for each model size (from 1 to the size of the selected model) and the subset is the best set of terms for that model size. (These subsets are specified in `$prune.terms` in `earth`'s return value.) Only subsets that are smaller than or equal in size to the final model are used for estimating variable importance.

2. The `rss` criterion first calculates the decrease in the RSS for each subset relative to the previous subset. (For multiple response models, RSS's are calculated over all responses.) Then for each variable it sums these decreases over all subsets that include the variable. Finally it scales the summed decreases so the maximum summed decrease is 100. Variables which cause larger net decreases in the RSS are considered more important.

3. The `gcv` criterion is the same, but uses the GCV instead of the RSS. Adding a variable can *increase* the GCV, i.e., adding the variable has a deleterious effect on the model. When this happens,

the variable could even have a negative total importance, and thus appear less important than unused variables.

Note that using RSq's and GRSq's instead of RSS's and GCV's would give identical estimates of variable importance. (RSq and GRSq are defined in the Value section of the [earth](#) help page.)

### Example

```
a <- earth(O3 ~ ., data=ozonel, degree=2)
evimp(a, trim=FALSE)
```

Yields the following matrix:

	col	used	nsubsets	gcv	rss
temp	4	1	10	100.00 1	100.00 1
humidity	3	1	8	12.65 1	14.77 1
ibt	7	1	8	12.65 1	14.77 1
doy	9	1	7	11.27 1	12.94 1
dpg	6	1	5	6.73 1	7.82 1
ibh	5	1	4	9.54 0	10.42 0
vis	8	1	4	4.34 1	5.26 1
wind	2	1	1	0.75 1	0.99 1
vh-unused	1	0	0	0.00 1	0.00 1

The rows are sorted on `nsubsets`. We see that `temp` is considered the most important variable, followed by `humidity`, and so on. We see that `vh` is unused in the final model, and thus is given an unused suffix and a 0 in the `used` column.

The `col` column gives the the column indices of the variables in the `x` argument to `earth` (after factors, if any, have been expanded; none in this example).

The `nsubsets` column is the number of subsets that included the corresponding variable. For example, `temp` appears in 10 subsets and `humidity` in 8.

The `gcv` and `rss` columns are scaled so the largest net decrease is 100.

The unnamed columns after the `gcv` and `rss` columns have a 0 if the corresponding criterion increases instead of decreasing (i.e., the ranking disagrees with the `nsubsets` ranking). We see that `ibh` is considered less important than `dpg` using the `nsubsets` criterion, but not with the `gcv` and `rss` criteria.

### Estimating variable importance in the MARS equation

Running `plotmo` with `ylim=NULL` (the default) gives an idea of which predictors in the MARS equation make the largest changes to the predicted value (but only with all other predictors at their median values).

Note that there is only a loose relationship between variable importance in the MARS equation and variable importance in the data — see the Introduction section above.

### Using `drop1` to estimate variable importance

As an alternative to `evimp`, you can use `drop1` (assuming you are using the formula interface to `earth`). Calling `drop1(my.earth.model)` will delete each predictor in turn from your model, rebuild the model from scratch each time, and calculate the GCV each time. You will get warnings that the `earth` library function `extractAIC.earth` is returning GCVs instead of AICs — but

that is what you want so you can ignore the warnings. (You can turn off just these warnings by passing `warn=FALSE` to `drop1`.) The column labeled AIC in the printed response from `drop1` will actually be a column of GCVs not AICs. The `Df` column is not much use in this context.

Remember that this technique only tells you how important a variable is with the other variables already in the model. It does not tell you the effect of a variable in isolation.

You will get lots of output from `drop1` if you built your original earth model with `trace>0`. You can set `trace=0` by updating your model before calling `drop1`. Do it like this:

```
my.model <- update.earth(my.model, trace=0).
```

### Estimating variable importance by building many models

The variance of the variable importances estimated from an earth model can be high (meaning that the estimates of variable importance in a model built with a different realization of the data would be different). This variance can be averaged out by building a bagged earth model and measuring variable importances in that (by taking the mean of the variable importances in the many earth models that make up the bagged model). You can do this easily using the functions `bagEarth` and `varImp` in the `caret` package.

Measuring variable importance using Random Forests is another way to go, independently of earth. See the functions `randomForest` and `importance` in the `randomForest` package.

### Remarks

This function is useful in practice but the following issues can make it misleading.

MARS models have a high variance — if the data changes a little, the set of basis terms created by the forward pass can change a lot. So estimates of predictor importance can be unreliable because they can vary with even slightly different training data.

Collinear (or otherwise related) variables can mask each other's importance, just as in linear models. This means that if two predictors are closely related, the forward pass will somewhat arbitrarily choose one over the other. The chosen predictor will incorrectly appear more important.

For interaction terms, each variable gets credit for the entire term — thus interaction terms are counted more than once and get a total higher weighting than additive terms (questionably). Each variable gets equal credit in interaction terms even though one variable in that term may be far more important than the other.

For factor predictors, importances are estimated on a per-level basis. The `evimp` function should probably aggregate these over all levels.

An example of conflicting importances (however, the results are fine with the default `pmethod`):

```
evimp(earth(mpg~., data=mtcars, pmethod="none"))
```

### Acknowledgment

Thanks to Max Kuhn for the original `evimp` code and for helpful discussions.

### See Also

`earth`, `plot.evimp`

### Examples

```
data(ozonel)
a <- earth(O3 ~ ., data=ozonel, degree=2)
ev <- evimp(a, trim=FALSE, sqrt.=TRUE)
```

```
plot(ev)
print(ev)
```

---

```
format.earth      Format "earth" objects
```

---

## Description

Return a string representing an [earth](#) expression.

## Usage

```
## S3 method for class 'earth':
format(x = stop("no 'x' arg"),
       digits = getOption("digits"), use.names = TRUE,
       decomp = "anova", style = "h", colon.char = ":", ...)
```

## Arguments

<code>x</code>	An <a href="#">earth</a> object. This is the only required argument.
<code>digits</code>	Number of significant digits. The default is <code>getOption("digits")</code> .
<code>use.names</code>	If TRUE (default), use variable names. Else use names of the form <code>x[, 1]</code> .
<code>decomp</code>	One of "anova" (default) order the terms using the "anova decomposition", i.e., in increasing order of interaction "none" order the terms as created during the earth forward pass.
<code>style</code>	Formatting style. One of "h" (default) more compact "pmax" for those who prefer it and for compatibility with old versions of earth "max" is the same as "pmax" but prints max rather than pmax "bf" basis function format.
<code>colon.char</code>	Change colons in the returned string to <code>colon.char</code> . Default is ":", i.e., no change. Specifying <code>colon.char="*"</code> can be useful in some contexts to change names of the form <code>x1:x2</code> to <code>x1*x2</code> .
<code>...</code>	Unused, but provided for generic/method consistency.

## Value

A character representation of the earth object.

If there are multiple responses, `format.earth` will return multiple strings.

If there are embedded GLM model(s), the strings for the GLM model(s) come after the strings for the standard earth model(s).

**Note**

The FAQ section for [earth](#) has some comments on the "anova" ordering.

Using `format.earth`, perhaps after hand editing the returned string, you can create an alternative to `predict.earth`. For example:

```
as.func <- function(object, digits = 8, use.names = FALSE, ...)
  eval(parse(text=paste(
    "function(x){\n",
    "if(is.vector(x))\n",
    "  x <- matrix(x, nrow = 1, ncol = length(x))\n",
    "with(as.data.frame(x),\n",
    "  format(object, digits = digits, use.names = use.names, style = "pmax", ...),
    ")\n",
    "})\n", sep = "")))
a <- earth(Volume ~ ., data = trees)
my.func <- as.func(a, use.names = FALSE)
my.func(c(10,80))      # yields 18.11
predict(a, c(10,80))  # yields 18.11, but is slower
```

The `earth` package also provides a function `format.lm`. It has arguments as follows

`format.lm(x, digits=getOption("digits"), use.names=TRUE, colon.char=":")`  
 (Strictly speaking, `format.lm` doesn't belong in the `earth` package.) Example:

```
a <- lm(Volume ~ Height*Girth, data = trees)
cat(format(a, colon.char="*"))

# yields:
#   69.4
#   - 1.30 * Height
#   - 5.86 * Girth
#   + 0.135 * Height*Girth
```

**See Also**

[earth](#), [pmax](#),

**Examples**

```
a <- earth(Volume ~ ., data = trees)
cat(format(a))

# yields:
#   27.2
#   + 6.18 * h(Girth-14)
#   - 3.27 * h(14-Girth)
#   + 0.491 * h(Height-72)

cat(format(a, style="pmax")) # default formatting style prior to earth version 1.4
```

```

# yields:
# 27.2
# + 6.18 * pmax(0, Girth - 14)
# - 3.27 * pmax(0, 14 - Girth)
# + 0.491 * pmax(0, Height - 72)

cat(format(a, style="bf"))

# yields:
# 27.2
# + 6.18 * bf1
# - 3.27 * bf2
# + 0.491 * bf3
#
# bf1 h(Girth-14)
# bf2 h(14-Girth)
# bf3 h(Height-72)

```

---

mars.to.earth	<i>Convert a "mars" object from the "mda" package to an "earth" object</i>
---------------	--

---

## Description

Convert a `mars` object from the "mda" package to an `earth` object

## Usage

```
mars.to.earth(object)
```

## Arguments

`object` A "mars" object, created using `mars` in the mda package.

## Value

The value is the same format as that returned by `earth` but with skeletal versions of `rss.per.subset`, `gcv.per.subset`, and `prune.terms`.

You can fully initialize these components by calling `update.earth` after `mars.to.earth`, but if you do this `selected.terms` may change. However with `pmethod="backward"` a change is unlikely — `selected.terms` would change only if GCVs are so close that numerical errors have an effect.

## Note

Perhaps the most notable difference between `mars` and `earth` objects is that `mars` returns the MARS basis matrix in a field called "x" whereas `earth` returns "bx" with only the selected terms. Also, `earth` returns "dirs" rather than "factors", and in `earth` this matrix can have entries

of value 2 for linear predictors. The calculation of minspan in `earth` follows Friedman's paper more closely and is slightly different from `mars`.

For details of other differences between `mars` and `earth` objects, see the comments in the source code of `mars.to.earth()`.

### Weights

Note that the `w` argument is actually ignored by `mars`. The equivalent `earth` argument `weights` is also not yet supported, and you will get a warning.

`mars` normalizes `wp` to (euclidean) length 1; `earth` normalizes `wp` to length equal to the number of responses, i.e., the number of columns in `y`. This change was made so an all 1s `wp` (or in fact any all constant `wp`) is equivalent to using no `wp`.

If the original call to `mars` used the `wp` argument, `mars.to.earth` will run `update.earth` to force consistency. This could modify the model, so a warning is issued.

### See Also

[earth](#), [mars](#)

### Examples

```
if (require(mda)) {
  a <- mars(trees[,-3], trees[,3])
  a <- mars.to.earth(a)
  summary(a, digits = 2) # the standard earth functions can now be used

  # yields (note the reconstructed call):
  #   Call: earth(x=trees[, -3], y=trees[, 3])
  #
  #
  #           Y
  # (Intercept) 19.76
  # h(Girth-12)  5.40
  # h(12-Girth) -2.56
  # h(Height-76) 0.72
  #
  # Selected 4 of 5 terms, and 2 of 2 predictors
  # Importance: Girth, Height
  # Number of terms at each degree of interaction: 1 3 (additive model)
  # GCV 13   RSS 251   GRSq 0.95   RSq 0.97
}
```

---

`model.matrix.earth` *Get the "earth" basis matrix*

---

### Description

Get the basis matrix of an [earth](#) object.

**Usage**

```
## S3 method for class 'earth':
model.matrix(object = stop("no 'object' arg"),
             x = NULL, subset = NULL, which.terms = NULL,
             ...,
             env = parent.frame(),
             trace = 0,
             Callers.name = "model.matrix.earth")
```

**Arguments**

object	An <a href="#">earth</a> object. This is the only required argument.
x	An input matrix with the same number of columns as the x matrix used to construct the original <a href="#">earth</a> object. Default is NULL, meaning use the original x matrix after taking the original subset, if any.
subset	Which rows to use in x. Default is NULL, meaning use all of x.
which.terms	Which terms to use. Default is NULL, meaning use object\$selected.terms.
...	Unused, but provided for generic/method consistency.
env	For internal use.
trace	Default 0. Set to non-zero to see which data model.matrix.earth is using.
Callers.name	For internal use (used by earth in trace messages).

**Value**

A basis matrix `bx` of the same form returned by [earth](#).

If `x`, `subset`, and `which.terms` are all NULL, this function returns the object's `bx`. In this case, it is perhaps easier to simply use `object$bx`.

The format of `bx` is described in [earth](#). The matrix `bx` can be used as the input matrix to `lm` or `glm`, as shown below in the example. In fact, that is what `earth` does internally after the pruning pass — it calls `lm.fit`, and additionally `glm` if `earth`'s `glm` argument is used.

**See Also**

[earth](#)

**Examples**

```
data(trees)
a <- earth(Volume ~ ., data = trees)
summary(a, decomp = "none") # "none" to print terms in same seq as a.lm below

# yields:
# Call: earth(formula=Volume~., data=trees)
#
#               coefficients
```

```

# (Intercept)          27.246
# h(Girth-14)          6.177
# h(14-Girth)         -3.266
# h(Height-72)        0.491
#
# Selected 4 of 6 terms, and 2 of 2 predictors
# Importance: Girth, Height
# Number of terms at each degree of interaction: 1 3 (additive model)
# GCV 10.6    RSS 197    GRSq 0.962    RSq 0.976

bx <- model.matrix(a)           # equivalent to bx <- a$bx
a.lm <- lm(trees$Volume ~ bx[,-1]) # -1 to drop intercept
summary(a.lm)                   # yields same coeffs as above summary
                                # displayed t values are not meaningful

# yields:
# Call:
# lm(formula = trees$Volume ~ bx[, -1])
#
# Residuals:
#    Min       1Q   Median       3Q      Max
# -4.882 -1.770  0.281  1.646  4.983
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept)    27.246     1.123   24.26 < 2e-16
# bx[, -1]h(Girth-14)  6.177     0.354   17.44 3.2e-16
# bx[, -1]h(14-Girth) -3.266     0.335   -9.76 2.4e-10
# bx[, -1]h(Height-72) 0.491     0.123    3.99 0.00045
#
# Residual standard error: 2.7 on 27 degrees of freedom
# Multiple R-squared:  0.976,    Adjusted R-squared:  0.973
# F-statistic: 361 on 3 and 27 DF,  p-value: <2e-16

```

---

ozone1

*Ozone readings in Los Angeles with incomplete cases removed*


---

## Description

Ozone readings in Los Angeles, with incomplete cases removed.

## Format

A data frame with 330 observations on 10 variables.

o3	daily maximum of the hourly average ozone concentrations in Upland, CA
vh	500 millibar pressure height, measured at the Vandenberg air force base
wind	wind speed in mph at LAX airport
humidity	humidity in percent at LAX
temp	Sandburg Air Force Base temperature in degrees Fahrenheit
ibh	temperature inversion base height in feet

dpg	pressure gradient from LAX to Daggert in mm Hg
ibt	inversion base temperature at LAX in degrees Fahrenheit
vis	visibility at LAX in miles
doy	day of the year

### Source

This data set was copied from `library(faraway)` and the name changed to `ozone1` to prevent a name clash. The data were originally made available by Leo Breiman who was a consultant on a project where the data were generated. Example analyses using these data may be found in Faraway and in Hastie and Tibshirani.

```
> ozone1
      O3   vh wind humidity temp  ibh dpg ibt vis doyear
1     3 5710   4      28   40 2693 -25  87 250   33
2     5 5700   3      37   45  590 -24 128 100   34
3     5 5760   3      51   54 1450  25 139  60   35
...
330  1 5550   4      85   39 5000   8  44 100 390
```

### References

Faraway (2005) *Extending the Linear Model with R* <http://www.maths.bath.ac.uk/~jjf23>  
Hastie and Tibshirani (1990) *Generalized Additive Models* <http://www-stat.stanford.edu/~hastie/pub.htm>

### See Also

[earth](#)

---

plot.earth                      *Plot an "earth" object*

---

### Description

Plot an `earth` object. The plot shows model selection, cumulative distribution of the residuals, residuals versus fitted values, and the residual QQ plot.

### Usage

```
## S3 method for class 'earth':
plot(x = stop("no 'x' arg"),
     which = 1:4, ycolumn = 1,
     caption = if(do.par) NULL else "",
     col.rsq = "lightblue", col.loess = col.rsq,
     col.qq = col.rsq, col.grid = "grey",
```

```
col.vline = "grey", lty.vline = 3,
col.legend = 1, col.npreds = 1,
nresiduals = 1000, cum.grid = "percentages", rlim = c(-1,-1),
jitter = 0, id.n = 3, labels.id = rownames(residuals(x, warn=FALSE)),
legend.pos = NULL, do.par = TRUE,
main = NULL, pch = 1, ...)
```

## Arguments

x	An <code>earth</code> object. This is the only required argument. (This argument is called "x" for consistency with the generic <code>plot</code> .)
which	Which plots to plot. Default is 1:4, meaning all. <ol style="list-style-type: none"> <li>1) model selection (GRSq plot)</li> <li>2) cumulative distribution of absolute values of residuals</li> <li>3) residuals versus fitted values</li> <li>4) QQ plot of residuals</li> </ol>
ycolumn	Specify which column of the response to plot if the model has multiple responses. Default is 1. This argument does not affect the model selection plot which is always across all responses. TODO there is an issue in the handling of <code>ycolumn</code> for multiple level factor responses. Does <code>ycolumn</code> refer to the column in the observed or predicted response?
caption	Overall caption. The default value is <code>if(do.par) NULL else ""</code> . Values are: <ul style="list-style-type: none"> <li>"string" string</li> <li>"" no caption</li> <li>NULL generate a caption from the <code>\$call</code> component of the <code>earth</code> object.</li> </ul> <i>For all the col arguments, 0 means don't plot the corresponding graph element.</i>
col.rsq	Color of RSq line in model selection plot. Default is "lightblue".
col.loess	Color of <code>loess</code> line in residuals plot. Default is <code>col.rsq</code> . Generating the loess line occasionally causes warnings such as "Warning: pseudoinverse used". To get rid of these warnings, set <code>col.loess=0</code>
col.qq	Color of QQ line. Default is <code>col.rsq</code> .
col.grid	Color of grid lines in cumulative distribution plot. Default is "grey".
col.vline	Color of vertical line at best model in model plot. Default is "grey".
lty.vline	Line type of vertical line at best model in model plot. Default is 3.
col.legend	Color of legend (inside plot area) of model plot. Default is 1, meaning draw a legend. Use 0 for no legend.
col.npreds	Color of the "number of predictors" plot within the model plot. Default is 1. Use 0 for no "number of predictors" plot.
nresiduals	Maximum number of residuals to plot. Use -1 for all. Default is 1000 (not all to reduce over-plotting). A systematic sample of size <code>nresiduals</code> is taken but the largest few residuals are always included.

cum.grid	Specify grid on cumulative distribution graph. Values are: "none" no grid on cumulative distribution plot "grid" add grid "percentages" (default) add grid and percentage labels to quantile lines.
rlim	Two element vector $c(\min, \max)$ specifying min and max values on the y axis of the RSq plot. Default is $c(-1, -1)$ . Special value $\min=-1$ means the minimum y axis value is the smallest GRSq or RSq value excluding the intercept values. Special value $\max=-1$ means the maximum y axis value is the largest GRSq or RSq value.
jitter	Jitter applied to GRSq and RSq values to minimize over-plotting. Default is 0, meaning no jitter. A typical useful value is 0.01.
id.n	Number of largest residuals to be labeled. Default is 3.
labels.id	Residual names. Default is <code>rownames(residuals(x))</code> .
legend.pos	NULL (default) means position legend automatically. Else specify $c(x, y)$ in user coordinates.
	<i>The following settings are related to <code>par()</code> and are included so you can override the defaults.</i>
do.par	Call <code>par()</code> for global settings as appropriate. Default is TRUE, which sets <code>mflow</code> , <code>mar=c(4, 4, 2, 1)</code> , <code>mgp=c(1.6, 0.6, 0)</code> , <code>cex=0.7</code> . Set to FALSE if you want to append figures to an existing plot.
main	Title of each plot. Default is NULL, meaning generate figure headings automatically.
pch	Plot character for QQ and residuals plot. Default is 1.
...	Extra arguments passed to plotting functions.

## Note

### Interpreting the plot.earth graphs

For concreteness, the description below is based on the graphs plotted by `example(plot.earth)`. The graphs plotted by `plot.earth`, apart from the Model Selection graph, are standard tools used in residual analysis and more information can be found in most linear regression textbooks.

One should be wary of over-interpretation of the graphs, since the residuals are measured on the training data rather than on new data. In linear models that is usually not an issue, but for flexible models like MARS the residuals measured on the training data give an optimistic view of the model's predictive ability.

### Nomenclature.

The *residuals* are the differences between the values predicted by the model and the corresponding response values. In this help page the residuals are all measured on the training data. The *residual sum of squares* (RSS) is the sum of the squared values of the residuals. *R-Squared* (RSq, also called the *coefficient of determination*) is a normalized form of the RSS, and, depending on the model, varies from 0 (a model that always predicts the same value i.e. the mean observed response value) to 1 (a model that perfectly predicts the responses in the training data). The *Generalized Cross*

*Validation* (GCV) is a form of the RSS penalized by the effective number of model parameters (and divided by the number of observations). More details can be found in the FAQ section of the [earth](#) help page. The *GRSq* normalizes the GCV in the same way that the *RSq* normalizes the RSS. The GCV and *GRSq* are measures of the generalization ability of the model, i.e., how well the model would predict using data not in the training set. There is some arbitrariness in their values since the effective number of model parameters is a just an estimate in MARS models.

In the example **Model Selection** graph, the *RSq* and *GRSq* run together at first, but diverge as the number of terms increases. This is typical behavior, and what we are seeing is an increased penalty being applied to the GCV as the number of model parameters increases. The vertical gray dotted line is positioned at the maximum *GRSq* and indicates that the best model has 11 terms and uses all 8 predictors (the number of predictors is shown by the black dotted line). The graph also shows the number of predictors and terms we would need if we were prepared to accept a lower *GRSq* (use the `earth` parameter `nprune` to trim the model).

The **Cumulative Distribution** graph shows the cumulative distribution of the absolute values of residuals. What we would ideally like to see is a graph that starts at 0 and shoots up quickly to 1. In the example graph, the median absolute residual is about 2.2 (look at the vertical gray line for 50%). We see that 95% of the absolute values of residuals are less than about 7.1 (look at the vertical gray line for 95%). So in the training data, 95% of the time the predicted value is within 7.1 units of the observed value.

The **Residuals vs Fitted** graph shows the residual for each value of the predicted response. By comparing the scales of the axes one can get an immediate idea of the size of the residuals relative to the predicted values.

Ideally the residuals should show constant variance i.e. the residuals should remain evenly spread out as the fitted values increase. However, in the example graph we see heteroscedasity — the residuals spread out in a "<" shape. There is a decrease in the accuracy of the predictions as the predicted value increases.

To reduce the heteroscedasity, we could refit the model after performing a transform on the response. A `log` transform, for instance, would even out the residuals:

```
a1 <- earth(log(O3) ~ ., data = ozone1, degree = 2)
plot(a1)
```

Transforming the data may cause other problems, such as mismatches to a known underlying physical model or difficulties in interpretation, so it's best to consult (or become) an expert on the type of data being modeled (in this case, ozone pollution data).

The pale blue line is a `loess` fit. (Readers not familiar with `loess` fits can think of them as fancy moving averages.) In this instance it shows that the mean residual is more or less constant except at low fitted values. The end effect is possibly due to failure of the model in that region because of smaller residuals, but cause and effect get tangled here. Compare the residuals of the `earth` model to the linear model, and notice how the pattern of residuals show that the `earth` model is more succesful at modeling non-linearities in the data:

```
a2 <- lm(O3 ~ ., data = ozone1)
plot(a2, which=1)
```

One should always eyeball the residuals themselves rather than blindly trusting the `loess` fit, which is itself an approximation. However, in our example `earth` model the `loess` line appears reliable.

Cases 192, 193, and 226 have the largest residuals and fall suspiciously into a separate cluster. (If overplotting makes the labels hard to read, reduce the number of labels with the `id.n` argument of `plot.earth`.) As a general rule, it is worthwhile investigating cases with large residuals. Perhaps they should be excluded when building the model. On the other hand, it is possible that they reveal something important about the data that could warrant changes to the model. In our example it is also worthwhile looking at cases with *small* residuals because of non-linearity in that region. To see the example input matrix ordered on the magnitude of the residuals, use `ozone1[order(abs(a$residuals)),]`.

Sometimes groups of residuals appear in a series of straight lines with slopes of -1. This effect is slightly visible in the example graph. These lines usually do not indicate a problem. They are formed when a set of plotted points has the same observed value, commonly due to discretization in the measurement of the observed response.

The **Normal Q-Q** graph compares the distribution of the residuals to a normal distribution. If the residuals are distributed normally they will lie on the line. Following R convention, the abscissa is the normal axis and the ordinate is the residual axis; some popular books have it the other way round. In our example, we see divergence from normality in the left tail — the left tail of the distribution is fatter than that of a normal distribution. Once again, we see that cases 192, 193, and 226 have the largest residuals.

#### About `plot.earth` and `earth-glm` models

"Earth-glm" models are models created with a `glm` argument to `earth`.

In `earth-glm` models, much of the analysis in the above section does not apply because in these models the residuals are not assumed to have a normal distribution.

Note that the residuals plotted by `plot.earth` are residuals from `earth`'s call to `lm` after the pruning pass, not `glm` residuals. That is, `plot.earth` ignores the `glm` part of the model, if any.

For `earth-glm` models, `plotd` can be useful.

#### Why doesn't `plot.earth` print GLM information?

It's just too much to display. You can instead call `plot` on the `glm.list` in the `earth` model like this:

```
data(etitanic)
a <- earth(survived ~ ., data=etitanic, glm=list(family=binomial))
par(mfrow=c(2,2))
plot(a$glm.list[[1]])
```

#### I want to add lines or points to the RSq plot, and am having trouble getting my axis scaling right. Help?

Use `do.par=FALSE`. With `do.par=FALSE`, the axis scales match the axis labels. With `do.par=TRUE`, `plot.earth` restores the `par` parameters and axis scales to what they were before calling `plot.earth`. This usually means that the x- and y-axis scales are both 0 to 1.

#### See Also

[earth](#), [plot.earth.models](#), [plotd](#), [plotmo](#)

## Examples

```
data(ozonel)
a <- earth(O3 ~ ., data = ozonel, degree = 2)
plot(a)
```

---

plot.earth.models *Compare "earth" models by plotting them.*

---

## Description

Compare `earth` models by plotting them.

## Usage

```
plot.earth.models(x = stop("no 'x' arg"), which = c(1:2),
  caption = "", rlim = c(0,1), jitter = 0,
  col.grsq = discrete.plot.cols(length(x)),
  col.rsq = 0, col.npreds = 0, col.cum = NULL,
  col.vline = "grey", lty.vline = 3, col.legend = 1,
  legend.pos = NULL, legend.text = NULL, do.par = TRUE,
  main = "Model Comparison", ...)
```

## Arguments

- |                      |   |
|----------------------|---|
| <code>x</code>       | A list of one or more <code>earth</code> objects, or a single <code>earth</code> object. This is the only required argument. (This argument is called 'x' for consistency with the generic <code>plot</code> .)   |
| <code>which</code>   | Which plots to plot: 1 model, 2 cumulative distribution of residuals. Default is 1:2, meaning both.   |
| <code>caption</code> | Overall caption. Values:<br>"string" string<br>"" (default) no caption<br>NULL generate a caption from the <code>\$call</code> component of the <code>earth</code> objects.   |
| <code>rlim</code>    | Two element vector <code>c(min, max)</code> specifying min and max values on the y axis of the RSq/GRSq plot. Default is <code>c(0, 1)</code> .<br>The special value <code>min=-1</code> means the minimum y axis value is the smallest GRSq or RSq, excluding the intercept.<br>The special value <code>max=-1</code> means the maximum y axis value is the largest GRSq or RSq. |
| <code>jitter</code>  | Jitter applied to GRSq and RSq values to minimize over-plotting. Default is 0, meaning no jitter. A typical useful value is 0.01.<br><br><i>For all the col arguments below, 0 means do not plot the corresponding graph element. You can use vectors of colors for the col arguments.</i>  |

col.grsq	Vector of colors for the GRSq plot. The default is <code>discrete.plot.cols(length(x))</code> which is vector of distinguishable colors, the first three of which are also distinguishable on a monochrome printer. You can examine the colors using <code>earth:::discrete.plot.c</code>
col.rsq	Vector of colors for the RSq plot. Default is 0, meaning no RSq plot.
col.npreds	Vector of colors for the "number of predictors" plot within the model selection plot. Default is 0, meaning no "number of predictors" plot. The special value <code>NULL</code> means borrow <code>col.grsq</code> (or <code>col.rsq</code> if <code>col.grsq</code> is <code>NULL</code> ).
col.cum	Vector of colors for the cumulative distribution plot. The special value <code>NULL</code> (default) means borrow <code>col.grsq</code> (or <code>col.rsq</code> if <code>col.grsq</code> is <code>NULL</code> ).
col.vline	A vertical line is drawn for each object to show which model size was chosen for that object. The color of the line is <code>col.vline</code> . Default is "grey".
lty.vline	Line type of vertical lines (a vertical line is drawn to show the selected model for each object). Can be a vector. Default is 3.
col.legend	Default is 1, meaning draw a legend. Use 0 for no legend. The legend is drawn in the cumulative distribution graph, if that graph is plotted. Else the legend is drawn in the model comparison chart.
legend.pos	The special value <code>NULL</code> (default) means position the legend automatically. Else specify <code>c(x, y)</code> in user coordinates.
legend.text	Vector of strings to use as legend text. The special value <code>NULL</code> (default) means generate the legend text automatically from <code>call\$formula</code> .
	<i>The following settings are related to <code>par()</code> and are included so you can override the defaults.</i>
do.par	Call <code>par()</code> for global settings as appropriate. Default is <code>TRUE</code> , which sets <code>mfrw</code> , <code>mar=c(4, 4, 2, 3)</code> , <code>mgp=c(1.6, 0.6, 0)</code> , <code>cex=0.7</code> . Set to <code>FALSE</code> if you want to append figures to an existing plot.
main	Title of each plot. Default is <code>NULL</code> , meaning generate figure headings automatically.
...	Extra arguments passed to plotting functions.

**Note**

This function ignores GLM and cross-validation components of the earth model, if any.

**See Also**

[earth](#), [plot.earth](#), [plot.earth.models](#), [plotd](#), [plotmo](#)

**Examples**

```
data(ozone1)
a1 <- earth(O3 ~ ., data = ozone1, degree = 2)
a2 <- earth(O3 ~ .-wind, data = ozone1, degree = 2, nk = 31)
a3 <- earth(O3 ~ .-humidity, data = ozone1, degree = 2, nk = 31)
plot.earth.models(list(a1, a2, a3), rlim=c(.6, .8))
```

---

plot.evimp                      *Plot an "evimp" object*

---

### Description

Plot an `evimp` object.

### Usage

```
## S3 method for class 'evimp':
plot(x = stop("no 'x' arg"),
     cex.var = 1,
     type.nsubsets = "l", col.nsubsets = "black", lty.nsubsets = 1,
     type.gcv = "l", col.gcv = "lightblue", lty.gcv = 1,
     type.rss = "l", col.rss = "gray60", lty.rss = 1,
     cex.legend = 1, x.legend = nrow(x), y.legend = x[1, "nsubsets"],
     main = "Variable importance", do.par=TRUE, ...)
```

### Arguments

<code>x</code>	An <code>evimp</code> object.
<code>cex.var</code>	<code>cex</code> for variable names. Default is 1. Make smaller (say 0.8) if you have lots of variables.
<code>type.nsubsets</code>	Plot type for nsubsets graph. Default is "l". Use "n" for none, "b" looks good too.
<code>col.nsubsets</code>	Color of nsubsets line. Default is "black".
<code>lty.nsubsets</code>	Line type of nsubsets line. Default is 1.
<code>type.gcv, col.gcv, lty.gcv</code>	As above but for the gcv plot
<code>type.rss, col.rss, lty.rss</code>	As above but for the rss plot
<code>cex.legend</code>	<code>cex</code> for legend strings. Default is 1. Make smaller (say 0.8) if you want a smaller legend.
<code>x.legend</code>	x position of legend. Use 0 for no legend.
<code>y.legend</code>	y position of legend.
<code>main</code>	Main title. Default is "Variable importance".
<code>do.par</code>	Call <code>par()</code> for global settings as appropriate. Default is <code>TRUE</code> , which sets <code>oma=c(bottom.margin, 0, 0, 3)</code> , <code>cex=cex.var</code> . Set to <code>FALSE</code> if you want to append figures to an existing plot.
<code>...</code>	Extra arguments passed to plotting functions.

### See Also

[earth](#), [evimp](#), [plot.earth.models](#), [plotmo](#)

**Examples**

```
data(ozonel)
a <- earth(O3 ~ ., data=ozonel, degree=2)
ev <- evimp(a)
plot(ev)
print(ev)
```

plotd

*Plot the distribution of predictions for each class***Description**

Draw a plot of the distribution of the predicted values for each class. Can be used for [earth](#) models, but also for models built by [lm](#), [glm](#), [lda](#), etc.

**Usage**

```
plotd(obj, hist = FALSE, type = "response", nresponse = NULL, dichot = FALSE,
      trace = FALSE, xlim = NULL, ylim = NULL, jitter = FALSE, main=NULL,
      xlab = "Predicted Value", ylab = if(hist) "Count" else "Density",
      lty = 1, col = c("grey70", 1, "lightblue", "brown", "pink", 2, 3, 4),
      fill = if(hist) col[1] else 0,
      breaks = "Sturges", labels = FALSE,
      kernel = "gaussian", adjust = 1, zero.line = FALSE,
      legend = TRUE, legend.names = NULL, legend.pos = NULL,
      legend.cex = .8, legend.bg = "white", legend.extra = FALSE,
      vline.col = 0, vline.thresh = .5, vline.lty = 1, vline.lwd = 1,
      err.thresh = vline.thresh, err.col = 0, err.border = 0, err.lwd = 1,
      xaxt = "s", yaxt = "s", xaxis.cex = 1, sd.thresh = 0.01, ...)
```

**Arguments**

To start off, look at the arguments `obj`, `hist`, `type`.  
 For predict methods with multiple column responses, see the `nresponse` argument.  
 For factor responses with more than two levels, see the `dichot` argument.

`obj` Model object. Typically a model which predicts a class or a class discriminant.

`hist` FALSE (default) to call [density](#) internally.  
 TRUE to call [hist](#) internally.

`type` Type parameter passed to [predict](#). Default is "response". See the predict method for your object for possible values; for example see [predict.glm](#) or [predict.earth](#). Typically you would set `hist=TRUE` if `type="class"`.

`nresponse` Column index for predicted responses with multiple columns. The default is NULL, meaning use all columns of the predicted response.

dichot	Dichotomise the predicted response. This argument is ignored except for models where the observed response is a factor with more than two levels and the predicted response is a numeric vector. The default FALSE separates the response into a group for each factor. With dichot=TRUE the response is separated into just two groups: the first level of the factor versus the remaining levels.
trace	Default FALSE. Use TRUE or 1 to trace plotd — useful to see how plotd partitions the predicted response into classes. Use 2 for a full dump of the internal matrices.
xlim	Limits of the x axis. The default NULL means determine these limits automatically, else specify c(xmin, xmax).
ylim	Limits of the y axis. The default NULL means determine these limits automatically, else specify c(ymin, ymax).
jitter	Jitter the histograms or densities horizontally to minimize overplotting. Default FALSE. Specify TRUE to automatically calculate the jitter, else specify a numeric jitter value.
main	Main title. Values: "string" string "" no title NULL (default) generate a title from the call.
xlab	x axis label. Default is "Predicted Value".
ylab	y axis label. Default is if(hist) "Count" else "Density".
lty	Per class line types for the plotted lines. Default is 1 (which gets recycled for all lines).
col	Per class line colors. The first few colors of the default are intended to be easily distinguishable on both color displays and monochrome printers.
fill	Fill color for the plot for the first class. For hist=FALSE, the default is 0, i.e., no fill. For hist=TRUE, the default is the first element in the col argument.
breaks	Passed to hist. Only used if hist=TRUE. Default is "Sturges". When type="class", setting breaks to a low number can be used to widen the histogram bars
labels	TRUE to draw counts on the hist plot. Only used if hist=TRUE. Default is FALSE.
kernel	Passed to density. Only used if hist=FALSE. Default is "gaussian".
adjust	Passed to density. Only used if hist=FALSE. Default is 1.
zero.line	Passed to plot.density. Only used if hist=FALSE. Default is FALSE.
legend	TRUE (default) to draw a legend, else FALSE.
legend.names	Class names in legend. The default NULL means determine these automatically.
legend.pos	Position of the legend. The default NULL means position the legend automatically, else specify c(x, y).
legend.cex	cex for legend. Default is .8.
legend.bg	bg color for legend. Default is "white".
legend.extra	Show (in the legend) the number of occurrences of each class. Default is FALSE.

<code>vline.thresh</code>	Horizontal position of optional vertical line. Default is 0.5. The vertical line is intended to indicate class separation. If you use this, don't forget to set <code>vline.col</code> .
<code>vline.col</code>	Color of vertical line. Default is 0, meaning no vertical line.
<code>vline.lty</code>	Line type of vertical line. Default is 1.
<code>vline.lwd</code>	Line width of vertical line. Default is 1.
<code>err.thresh</code>	x axis value specifying the error shading threshold. See <code>err.col</code> . Default is <code>vline.thresh</code> .
<code>err.col</code>	Specify up to three colors to shade the "error areas" of the density plot. The default is 0, meaning no error shading. This argument is ignored unless <code>hist=FALSE</code> . If there are more than two classes, <code>err.col</code> uses only the first two. This argument is best explained by running an example: <pre>data(etitanic) earth.model &lt;- earth(survived ~ ., data=etitanic) plotd(earth.model, vline.col=1, err.col=c(2,3,4))</pre> <p>The three areas are (i) the error area to the left of the threshold, (ii) the error area to the right of the threshold, and, (iii) the reducible error area. If less than three values are specified, <code>plotd</code> re-uses values in a sensible manner. Use values of 0 to skip areas. Disjoint regions are not handled well by the current implementation.</p>
<code>err.border</code>	Borders around the error shading. Default is 0, meaning no borders, else specify up to three colors.
<code>err.lwd</code>	Line widths of borders of the error shading. Default is 1, else specify up to three line widths.
<code>xaxt</code>	Default is "s". Use <code>xaxt="n"</code> for no x axis.
<code>yaxt</code>	Default is "s". Use <code>yaxt="n"</code> for no y axis.
<code>xaxis.cex</code>	Only used if <code>hist=TRUE</code> and <code>type="class"</code> . Specify size of class labels drawn on the x axis. Default is 1.
<code>sd.thresh</code>	Minimum acceptable standard deviation for a density. Default is 0.01. Densities with a standard deviation less than <code>sd.thresh</code> will not be plotted (a warning will be issued and the legend will say "not plotted").
<code>...</code>	Extra arguments passed to the predict method for the object.

### Note

This function calls `predict` with the data originally used to build the model, and with the `type` specified above. It then separates the predicted values into classes, where the class for each predicted value is determined by the class of the observed response. Finally, it calls `density` (or `hist` if `hist=TRUE`) for each class-specific set of values, and plots the results.

This function estimates distributions with the `density` and `hist` functions, and also calls `plot.density` and `plot.histogram`. For an overview see Venables and Ripley MASS section 5.6.

### Partitioning the response into classes

Considerable effort is made to partition the predicted response into classes in a sensible way. This is not always possible for multiple column responses and the `nresponse` argument should be used where necessary. The partitioning details depend on the types and numbers of columns in the observed and predicted responses. These in turn depend on the model object and the `type` argument.

Use the `trace` argument to see how `plotd` partitions the response for your model.

### Degenerate densities

A message such as

```
Warning: standard deviation of "male" density is 0, density is degenerate?
means that the density for that class will not be plotted (the legend will say "not plotted").
```

Set `sd.thresh=0` to get rid of this check, but be aware that histograms (and sometimes x axis labels) for degenerate densities will be misleading.

### Using plotd for various models

This function is included in the `earth` package but can also be used with other models.

Example with `glm`:

```
library(earth); data(etitanic)
glm.model <- glm(sex ~ ., data=etitanic, family=binomial)
plotd(glm.model)
```

Example with `lm`:

```
library(earth); data(etitanic)
lm.model <- lm(as.numeric(sex) ~ ., data=etitanic)
plotd(lm.model)
```

### Using plotd with lda

The `plotd` function has special handling for `lda` objects. For such objects, the `type` argument can take one of the following values:

```
"response" (default) linear discriminant
"ld" same as "response"
"class" predicted classes
"posterior" posterior probabilities
```

Example:

```
library(MASS); library(earth); data(etitanic)
lda.model <- lda(sex ~ ., data=etitanic)
plotd(lda.model) # linear discriminant by default
plotd(lda.model, type="class", hist=TRUE, labels=TRUE)
```

This handling of `type` is handled internally by `plotd` and `type` is not passed to `predict.lda` (`type` is used merely to select fields in the list returned by `predict.lda`). The `type` names can be abbreviated down to a single character.

For objects created with `lda.matrix` (as opposed to `lda.formula`), `plotd` blindly assumes that the `grouping` argument was the second argument.

`plotd` does not yet support objects created with `lda.data.frame`.

For `lda` responses with more than two factor levels, use the `nresponse` argument to select a column in the predicted response. Thus with the default `type="response"`, use `nresponse=1` to select just the first linear discriminant. The default `nresponse=NULL` selects all columns, which is typically not what you want for `lda` models. Example:

```
library(MASS); library(earth);
set.seed(1)      # optional, for reproducibility
example(lda)    # creates a model called "z"
plot(z, dimen=1) # invokes plot.lda from the MASS package
plotd(z, nresponse=1, hist=1) # equivalent using plotd
                                # nresponse=1 selects first linear discr.
```

The `dichot=TRUE` argument is also useful for `lda` responses with more than two factor levels.

### TODO

Handle degenerate densities in a more useful way.  
Add `freq` argument for `hist`.

### See Also

[density](#), [plot.density](#)  
[hist](#), [plot.histogram](#)  
[earth](#), [plot.earth](#), [plotmo](#)

### Examples

```
old.par <- par(no.readonly=TRUE); par(mfrow=c(2,2));
par(mar=c(4, 3, 1.7, 0.5)); par(mgp=c(1.6, 0.6, 0)); par(cex = 0.8)
data(etitanic)
fit <- earth(survived ~ ., data=etitanic, degree=2, glm=list(family=binomial))

plotd(fit)

plotd(fit, hist=TRUE, legend.pos=c(.25,220))

plotd(fit, hist=TRUE, type="class", labels=TRUE, xlab="", xaxis.cex=.8)

par(old.par)
```

plotmo

*Plot the predicted model response***Description**

Plot the predicted model response when varying one or two predictors while holding all the other predictors constant.

**Usage**

```
plotmo(object = stop("no 'object' arg"),
       degree1 = TRUE, degree2 = TRUE, ycolumn = 1, type="response",
       caption = if(do.par) NULL else "",
       ylim = NULL, clip = TRUE, inverse.func = NULL,
       col.response = 0, pch.response = 1, trace = FALSE,
       grid.func = median, grid.levels = NULL,
       ndegree1 = 500, lty.degree1 = 1, col.degree1 = 1,
       se = 0, col.shade = "lightblue", col.se = 0, lty.se = 2,
       func = NULL, col.func = "pink", pch.func = 20, nrug = 0,
       type2 = "persp", ngrid = 20,
       col.persp = "lightblue", col.image = grey(0:9/10),
       do.par = TRUE, main = NULL, theta = NA, phi = 30, shade = 0.5,
       ticktype = "simple", xlab = "", ylab = "", cex = NULL, ...)
```

**Arguments**

To start off, look at the arguments `object`, `degree1`, `degree2`, `ylim`, `clip`, and `do.par`. You will also need the `ycolumn` argument if your model has multiple responses.

Model object. This is the only required argument.

<code>degree1</code>	Index vector specifying main effect plots to include. Default is <code>TRUE</code> , meaning all. Perhaps the easiest way to use this argument (and <code>degree2</code> ) is to first plot all figures to see how the figures are numbered, then replot, using <code>degree1</code> to select the figures you want.
<code>degree2</code>	Index vector specifying interaction plots to include. Default is <code>TRUE</code> , meaning all.
<code>ycolumn</code>	Specify which column of the response to plot if the model has multiple responses. Default is 1.
<code>type</code>	type parameter passed onto <code>predict</code> . Default is "response". See <code>predict.earth</code> for other values for earth models.
<code>caption</code>	Overall caption. The default value is <code>if(do.par) NULL else ""</code> . Values are: "string" string "" no caption NULL generate a caption from <code>object\$call</code> and the response name.

<code>trace</code>	Set to TRUE to trace operation. Useful for tracking down error messages. Default is FALSE.
<code>ylim</code>	<p>Values are:</p> <p>NULL (default) all y axes have same limits (where "y" is actually "z" on degree2 plots). The limits are the min and max values of y across all (degree1 and degree2) plots. If <code>col.response != 0</code> then the original response is also included in the min and max calculation.</p> <p>NA each graph has its own y limits.</p> <p><code>c(ymin, ymax)</code> all graphs have the specified y limits.</p>
<code>clip</code>	Plot only values that are in the range of the response of the original data. Default is TRUE.
<code>inverse.func</code>	Function applied to the predicted response before plotting. Default is NULL, meaning do not apply a function. For example, you could use <code>inverse.func=exp</code> if your model formula is $\log(y) \sim x$ . Note, however, that is usually unnecessary to use <code>inverse.func</code> for models such as <code>glm</code> with <code>family=binomial</code> , because the plotted response is already a probability.
<code>col.response</code>	Color of response values (actually, the response sites in degree2 plots). Here "response" refers to the original data used to build the model. Default is 0, meaning don't plot the response. Ignored by <code>persp</code> plots.
<code>pch.response</code>	Plot character for <code>col.response</code> . Default is 1.
<code>grid.func</code>	<p>Function applied to x columns to calculate the values for numeric predictors not on the axes. (For factor predictors, see the <code>grid.levels</code> argument below.) Default is <code>median</code>. Examples:</p> <pre>plotmo(a, grid.func = mean) # mean instead of median grid.func &lt;- function(x) quantile(x)[2] # 25% quantile plotmo(a, grid.func = grid.func)</pre>
<code>grid.levels</code>	<p>This argument only applies if model x-matrix has factors. It has the same purpose as the <code>grid.func</code> argument above but applies to factor predictors, not numeric predictors. It is a list specifying which factor level to use for factor predictors not on the axes. If specified, <code>plotmo</code> will print messages reminding you of how the levels are set. Default is NULL meaning use the first level of each factor.</p> <pre>a &lt;- earth(survived ~ pclass+sex+age, data=etitanic, degree=2) plotmo(a, degree2=0, grid.levels=list(pclass="2nd", sex="male"))</pre> <p><b>The following arguments are for degree1 (main effect) plots</b></p>
<code>ndegree1</code>	Maximum number of points in each degree1 plot. Default is 500. Special value -1 means use <code>nrow(x)</code> .
<code>lty.degree1</code>	Line type of degree 1 plots. Default is 1.
<code>col.degree1</code>	Color of degree 1 plots. Default is 1.

<code>n rug</code>	Number of points in (jittered) rug. Default is 0 meaning no rug. Special value -1 for all, i.e., <code>n row(x)</code> .
<code>se</code>	Draw standard error bands at plus and minus <code>se</code> times the pointwise standard errors. Default is 0, meaning no standard error bands. A common value is 2. The <code>predict</code> method for <code>object</code> must be callable with <code>se.fit=TRUE</code> (examples are <code>predict.lm</code> but not <code>predict.earth</code> ).
<code>col.shade</code>	Color of <code>se</code> shading. Default is "lightblue". Set to 0 for no shading.
<code>col.se</code>	Color of <code>se</code> lines. Default is 0 meaning no lines just shading.
<code>lty.se</code>	Line type of <code>se</code> lines. Default is 2.
<code>func</code>	Superimpose <code>func(x)</code> if <code>func</code> is not NULL. Default is NULL. This is useful if you are comparing the model to a known function. Note that <code>func</code> is called with a single argument which is a dataframe with columns in the same order as the predictors in the <code>formula</code> or <code>x</code> used to build the model. Set <code>trace=TRUE</code> to see the column names and first few rows of this dataframe.
<code>col.func</code>	Color of <code>func</code> points. Default is "pink".
<code>pch.func</code>	Plot character for <code>func</code> points. Default is 20.

**The following arguments are for degree2 plots**

<code>type2</code>	Degree2 plot type. One of "persp" (default), "contour", or "image".
<code>col.persp</code>	Color of <code>persp</code> surface. Default is "lightblue". Set to 0 for no color.
<code>col.image</code>	Colors of <code>image</code> plot. Default is <code>grey(0:9/10)</code> . The default excludes <code>grey(1)</code> because that is the "color" of clipped values, see <code>clip</code> .
<code>ngrid</code>	Grid side length for degree2 plots. Default is 20.

**The following settings are related to `par()` and are included so you can override the defaults.**

<code>do.par</code>	Call <code>par()</code> for global settings as appropriate. Default is TRUE. Set to FALSE if you want to append figures to an existing plot.
<code>main</code>	Title of each plot. Default is NULL, meaning generate figure headings automatically.
<code>theta</code>	Rotation argument for <code>persp</code> . Default is NA, meaning automatically rotate each graph so the highest corner is furthest away. Use <code>theta=-45</code> for x and y increasing as you move into the paper.
<code>phi</code>	Phi argument for <code>persp</code> . Default is 30.
<code>shade</code>	Shade arg for <code>persp</code> . Default is 0.5.
<code>ticktype</code>	Ticks on <code>persp</code> plot. One of <code>simple</code> or <code>detailed</code> . Default is "simple".
<code>xlab</code>	Horizontal axis label on degree1 plots (for degree2 plots the abscissa labels are always the predictor names). Default is "", meaning none, which gives more plottable area. Set to NULL to use the predictor names as labels. If you use NULL, you may want to set <code>main=""</code> to avoid redundant labeling.
<code>ylab</code>	Vertical axis label. Default is "", meaning none, which gives more plottable area.

<code>cex</code>	Character expansion.
<code>...</code>	Extra arguments passed to plotting functions called by <code>plotmo</code> . Using arguments here may cause warnings which can often be safely ignored.

## Details

`Plotmo` is a general purpose model plotting function (but comes with the `earth` package). It is intended for models with quantitative responses.

### The general idea

`Plotmo` plots a degree1 plot of predicted values by changing one predictor while holding all other predictors at a constant value. For degree2 plots, `plotmo` changes two variables while holding all other predictors at a constant value.

The question arises: what should the constant be for each predictor? For numerical predictors it is the median of the predictor values in the input matrix `x`; for factor predictors it is the first level of the factor. You can change those defaults with the `grid.func` and `grid.levels` arguments.

Each graph shows only a thin slice of the data because most variables are fixed. You should be aware of that when interpreting the graph.

### Details of operation

Let's say the model `object` has three predictors, `x1`, `x2`, and `x3` (all numeric) and `plotmo` is about to plot the degree1 plot for `x2`. `Plotmo` first builds an input matrix with `ndegree1` rows and with column names `x1`, `x2`, and `x3`. It sets all entries for the `x1` column to `x1`'s median value (actually, the value returned by `grid.func` applied to `x1`). Likewise for the `x3` column. It sets the `x2` column to an equally spaced sequence of values from `min(x2)` to `max(x2)`. Finally, it calls `predict(type=type)` with the newly created input matrix, and plots the predicted values against the sequence of `x2` values.

Operation is similar for degree2 plots: all columns of the input matrix for `predict` are set to their medians except for the columns of the two predictors being plotted.

Note that by default `plotmo` calls `predict` with new data and `type="response"`, whereas `termplot` calls `predict` with `type="terms"`.

### Limitations

NAs are not allowed. To prevent confusing error messages from `plotmo`, remove NAs before building your model. (To be safe, you can use `na.action=na.fail` when building your model so you get an error message if you inadvertently have a NA.)

Weights are currently ignored, with a warning.

Factor predictors are not supported on degree2 graphs (you will get a reminder message).

To avoid confusing error messages from `plotmo`, keep the original formula you used to build the model simple. By default (i.e., when using `get.plotmo.x.default` and `get.pairs.default`), `plotmo` parses the input `formula` using `gsub`. This crude approach is not infallible but works for the common formulas. It determines which predictors are paired by looking for forms such as "`x1:x2`" or "`x1*x2`" in the model formula.

Variable names containing `\$` are not supported. The work around is to build the model using temporary variables or to use `attach`.

Plotmo can get confused by predictors in formulas which use indexing, such as `x[,3]`. The symptom is usually a message along the lines

```
Error in model.frame:  invalid type (list) for variable 'x[,3]'.
```

A message like

```
Warning in model.frame.default:  'newdata' had 50 rows but variable(s)
found have 31 rows
```

means that `model.frame.default` cannot find all the variables in the data frame created by `plotmo`.

### Minimum Requirements

Plotmo requires the following of the model object. These requirements are for default operation, which can be changed as described in the next section.

- 1) object must have a `predict` method that supports `type=response`.
- 2) object must have the following two components (which are searched for in the order given for each):
  - `$x`, or `$call$formula` (formula is required for degree2 plots), or `$call$x`.
  - `$y`, or `$call$formula`, or `$call$y`.
- 3) for optional standard error bands (see the `se` argument), object must have a `predict` method that can be called with `se.fit=TRUE`.

### Extending plotmo

Plotmo calls the following generic functions, all defined in the file `plotmo.R`:

```
plotmo.prolog
get.plotmo.x
get.plotmo.y
plotmo.predict
get.singles
get.pairs
```

Thus `plotmo` can be extended by writing new method functions, although the default functions may suffice for your object's class. See the source comments for details.

### FAQ

I want to add lines or points to a plot created by `plotmo`. and am having trouble getting my axis scaling right. Help?

Use `do.par=FALSE`. With `do.par=FALSE`, the axis scales match the axis labels. With `do.par=TRUE`, `plot.earth` restores the `par` parameters and axis scales to what they were before calling `plot.earth`. This usually means that the x- and y-axis scales are both 0 to 1.

### Author(s)

Stephen Milborrow. Users are encouraged to send feedback — use `milboATsonicPERIODnet` <http://www.milbo.users.sonic.net>.

### See Also

[termplot](#), [plot.earth](#), [plot.earth.models](#), [plotd](#)

**Examples**

```

data(ozonel)
a <- earth(O3 ~ ., data = ozonel, degree = 2)
plotmo(a)

## Not run:
# example with some arguments

plotmo(a, caption = "example", ylim = NULL, degree1 = c(1,2,4),
       degree2 = 4, col.response = 3, clip = FALSE, ticktype = "d", theta = -30)

# examples using functions other than earth (the models are more or less bogus)

plotmo(lm(O3 ~ log(temp) + humidity*temp, data=ozonel), se=2)

library(gam)
data(airquality)
airquality <- na.omit(airquality) # plotmo doesn't know how to deal with NAs
plotmo(gam(Ozone^(1/3) ~ lo(Solar.R) + lo(Wind, Temp), data = airquality))

library(mgcv)
plotmo(gam(O3 ~ s(doy) + s(humidity,temp), data=ozonel), se=2, ylim=NA)

## End(Not run)

```

---

predict.earth      *Predict with an "earth" model*

---

**Description**

Predict with an [earth](#) model.

**Usage**

```

## S3 method for class 'earth':
predict(object = stop("no 'object' arg"), newdata = NULL,
       type = c("link", "response", "earth", "class", "terms"),
       thresh = .5, trace = FALSE, ...)

```

**Arguments**

object	An <a href="#">earth</a> object. This is the only required argument.
newdata	Make predictions using newdata, which can be a dataframe, a matrix, or a vector with length equal to a multiple of the row length of the original input matrix $x$ . Note that this is more flexible than the predict methods for most R models. Default is NULL, meaning return values predicted from the training set.

type	Type of prediction. One of "link" (default), "response", "earth", "class", or "terms". See the <b>Note</b> below.
thresh	Threshold, a value between 0 and 1 when predicting a probability. Only applies when type="class". Default is .5. See the <b>Note</b> below.
trace	Default FALSE. Set to TRUE to see which data, subset, etc. predict.earth is using.
...	Unused, but provided for generic/method consistency.

### Value

The predicted values (a matrix for multiple response models). Except if type="terms", then a matrix with each column showing the contribution of a predictor.

### Note

#### Predicting with standard earth models

Use the default type="link", or possibly type="class".

Actually, the "link", "response", and "earth" choices all return the same value unless the glm argument was used in the original call to [earth](#).

#### Predicting with earth-GLM models

This section applies to earth models with a GLM component, i.e., when the glm argument was used in the original call to [earth](#).

The "link" and "response" options: see [predict.glm](#) for a description of these. In brief: for logistic models use type="link" to get log-odds and type="response" to get probabilities.

Use option "earth" to get the linear fit (this gives the prediction you would get if your original call to earth had no glm argument).

#### Predicting with "class"

Use option "class" to get the predicted class. With option "class", this function first makes predictions with type="response" and then assigns the predicted values to classes as follows:

- (i) When  $y$  is a *logical*, predict TRUE if the predicted probability is greater than thresh.
- (ii) When  $y$  is a *numeric*, predict TRUE if the predicted value is greater than thresh. Actually, this is identical to the above case, although thresh here may legitimately be a value outside the 0...1 range.
- (iii) When  $y$  is a *two level factor*, predict the second level if its probability is more than thresh. In other words, with the default thresh=.5 predict the most probable level.
- (iv) When  $y$  is a *three or more level factor*, predict the most probable level (and thresh is ignored).

#### Predicting with "terms"

The "terms" option returns a "link" response suitable for [termplot](#). Only the additive terms and the first response (for multi-response models) are returned. Also, "terms" always returns the earth terms, and ignores the GLM component of the model, if any.

**See Also**

[earth](#), [predict](#)

**Examples**

```
data(trees)
a <- earth(Volume ~ ., data = trees)
predict(a) # same as a$fitted.values
predict(a, c(10,80)) # yields 18.11
```

---

print.evimp	<i>Print an "evimp" object</i>
-------------	--------------------------------

---

**Description**

Print an [evimp](#) object. Just a simple utility that drops the class field when printing.

**Usage**

```
## S3 method for class 'evimp':
print(x = stop("no 'x' arg"), ...)
```

**Arguments**

x	An <a href="#">evimp</a> object.
...	Extra arguments passed to print functions.

**See Also**

[earth](#), [evimp](#)

**Examples**

```
data(ozonel)
a <- earth(O3 ~ ., data=ozonel, degree=2)
evimp(a) # implicitly calls print.evimp
```

---

residuals.earth      *Residuals for an "earth" model*

---

## Description

Residuals for an `earth` model.

## Usage

```
## S3 method for class 'earth':
residuals(object = stop("no 'object' arg"),
          type = NULL, warn = TRUE, ...)

## S3 method for class 'earth':
resid(object = stop("no 'object' arg"),
      type = NULL, warn = TRUE, ...)
```

## Arguments

<code>object</code>	An <code>earth</code> object. This is the only required argument.
<code>type</code>	One of " <code>earth</code> " (default) return earth residuals (from the <code>lm</code> fit on <code>bx</code> ) " <code>deviance</code> " Return the earth <code>lm</code> residuals unless the object has a <code>glm</code> component, in which case return the <code>glm</code> deviance residuals. " <code>pearson</code> " " <code>working</code> " " <code>response</code> " " <code>partial</code> " Return the corresponding <code>glm</code> residuals (from the <code>glm</code> fit on <code>bx</code> ). Can only be used if the earth model has a <code>glm</code> component.
<code>warn</code>	This function gives warnings when the results are not what you may expect. Use <code>warn=FALSE</code> to turn off just these warnings.
<code>...</code>	Unused, but provided for generic/method consistency.

## Value

The residual values (will be a matrix for multiple response models).

## See Also

`earth`  
`residuals`  
`resid` identical to `residuals`

**Examples**

```
data(etitanic)
a <- earth(pclass ~ ., data=etitanic, glm=list(family=binomial))
head(resid(a, warn=FALSE))      # earth residuals, a column for each response
head(resid(a, type="earth"))    # same
head(resid(a, type="deviance")) # GLM deviance residuals, a column for each response
```

---

summary.earth	<i>Summary method for "earth" objects</i>
---------------	---

---

**Description**

Summary method for [earth](#) objects.

**Usage**

```
## S3 method for class 'earth':
summary(object = stop("no 'object' arg"),
        details = FALSE, decomp = "anova",
        style = c("h", "pmax", "max", "bf"),
        digits = getOption("digits"), fixed.point=TRUE, ...)

## S3 method for class 'summary.earth':
print(x = stop("no 'x' arg"),
      details = x$details, decomp = x$decomp,
      digits = x$digits, fixed.point = x$fixed.point, ...)
```

**Arguments**

object	An <a href="#">earth</a> object. This is the only required argument for <code>summary.earth</code> .
x	A <a href="#">summary.earth</a> object. This is the only required argument for <code>print.summary.earth</code> .
details	Default is FALSE. Use TRUE to print more information about <a href="#">earth-glm</a> models. But note that the displayed P-values of the GLM coefficients are meaningless because of the amount of preprocessing by earth to select the regression terms.
decomp	Specify how terms are ordered. Default is "anova". Use "none" to order the terms as created by the forward.pass. See <a href="#">format.earth</a> for a full description.
style	Formatting style. One of " h " (default) more compact " pmax " for those who prefer it and for compatibility with old versions of earth " max " is the same as " pmax " but prints max rather than pmax " bf " basis function format.
digits	The number of significant digits. For <code>summary.earth</code> , the default is <code>getOption("digits")</code> . For <code>print.summary.earth</code> , the default is the <code>\$digits</code> component of object.

`fixed.point` Method of printing numbers in matrices. Default is TRUE which prints like this (making it easier to compare coefficients):

```
(Intercept)    15.029
h(temp-58)     0.313
h(234-ibt)    -0.046
...
```

whereas `fixed.point=FALSE` prints like this (which is more usual in R):

```
(Intercept)    1.5e+01
h(temp-58)     3.1e-01
h(234-ibt)    -4.6e-02
...
```

Matrices with two or fewer rows are never printed with a fixed point.

... Extra arguments are passed to `format.earth`.

### Value

The value is the same as that returned by `earth` but with the following extra components.

`strings` String(s) created by `format.earth`. For multiple response models, a vector of strings.

`digits`

`details`

`decomp`

`fixed.point` The corresponding arguments, passed on to `print.summary.earth`.

### Note

The printed Estimated importance uses `evimp` with the `nsubsets` criterion. The most important predictor is printed first, and so on.

### See Also

`earth`, `evimp`, `format.earth`

### Examples

```
a <- earth(Volume~ ., data = trees)
summary(a, digits = 2)

# yields:
# Call: earth(formula=Volume~., data=trees)
#
#               coefficients
# (Intercept)    27.25
# h(Girth-14)     6.18
# h(14-Girth)    -3.27
# h(Height-72)   0.49
```

```
#
#   Selected 4 of 6 terms, and 2 of 2 predictors
#   Importance: Girth, Height
#   Number of terms at each degree of interaction: 1 3 (additive model)
#   GCV 11   RSS 197   GRSq 0.96   RSq 0.98
```

---

```
update.earth      Update an "earth" model
```

---

## Description

Update an `earth` model.

## Usage

```
## S3 method for class 'earth':
update(object = stop("no 'object' arg"),
       formula. = NULL, ponly = FALSE, ..., evaluate = TRUE)
```

## Arguments

<code>object</code>	The earth object
<code>formula.</code>	The <code>formula.</code> argument is treated like <code>earth</code> 's <code>formula</code> argument.
<code>ponly</code>	Force pruning only, no forward pass. Default is <code>FALSE</code> , meaning <code>update.earth</code> decides automatically if a forward pass is needed. See note below.
<code>...</code>	Arguments passed on to <code>earth</code> .
<code>evaluate</code>	If <code>TRUE</code> (default) evaluate the new call, else return the call. Mostly for compatibility with the generic <code>update</code> .

## Value

The value is the same as that returned by `earth`. If `object` is the only parameter then no changes are made — the returned value will be the same as the original `object`.

## Note

If only the following arguments are used, a forward pass is unnecessary, and `update.earth` will perform only the pruning pass. This is usually much faster for large models.

```
object
glm
trace
nprune
pmethod
Get.crit
Eval.model.subsets
```

```
Print.pruning.pass
Force.txt.prune
Use.beta.cache
```

This automatic determination to do a forward pass can be overridden with the `ponly` argument. If `ponly=TRUE` the forward pass will be skipped and only the pruning pass will be executed. This is useful for doing a pruning pass with new data, typically with `penalty=0`. A similar use of `ponly=TRUE` is to do subset selection with a different `penalty` from that used to build the original model. With `trace=1`, `update.earth` will tell you if `earth`'s forward pass was skipped.

If you used `keepxy=TRUE` in your original call to `earth`, then `update.earth` will use the saved values of `x`, `y`, etc., unless you specify otherwise by arguments to `update.earth`. It can be helpful to set `trace=1` to see which `x`, etc. `update.earth` uses.

### See Also

[earth](#)

### Examples

```
data(ozonel)
(a <- earth(O3 ~ ., data = ozonel, degree = 2))

# yields:
#   Selected 11 of 21 terms, and 8 of 9 predictors
#   Importance: temp, humidity, ibt, doy, dpq, ibh, vis, wind, vh-unused
#   Number of terms at each degree of interaction: 1 5 5
#   GCV 13.5    RSS 3768    GRSq 0.791    RSq 0.822

update(a, formula = O3 ~ . - temp) # requires forward pass and pruning

# yields:
#   Selected 14 of 21 terms, and 8 of 8 predictors
#   Importance: ibt, humidity, doy, vh, ibh, dpq, wind, vis
#   Number of terms at each degree of interaction: 1 5 8
#   GCV 13.1    RSS 3499    GRSq 0.796    RSq 0.834

update(a, nprune = 8) # requires only pruning

# yields:
#   Selected 8 of 21 terms, and 6 of 9 predictors
#   Importance: temp, humidity, ibt, doy, dpq, ibh-unused, vis, vh-unused, wind-unused
#   Number of terms at each degree of interaction: 1 5 2
#   GCV 15.1    RSS 4435    GRSq 0.766    RSq 0.79

update(a, penalty=1, ponly=TRUE) # pruning pass only with a new penalty

# yields:
#   Selected 13 of 21 terms, and 8 of 9 predictors
#   Importance: temp, humidity, ibt, doy, dpq, ibh, vis, wind, vh-unused
#   Number of terms at each degree of interaction: 1 5 7
#   GCV 12.5    RSS 3671    GRSq 0.805    RSq 0.826
```

# Index

- \*Topic **datasets**
  - etitanic, 27
  - ozonel, 37
- \*Topic **models**
  - contr.earth.response, 1
  - earth, 2
  - evimp, 28
  - format.earth, 32
  - mars.to.earth, 34
  - model.matrix.earth, 36
  - plot.earth, 39
  - plot.earth.models, 43
  - plot.evimp, 45
  - plotd, 46
  - plotmo, 51
  - predict.earth, 56
  - print.evimp, 58
  - residuals.earth, 59
  - summary.earth, 60
  - update.earth, 62
- \*Topic **regression**
  - earth, 2
  - plotmo, 51
- \*Topic **smooth**
  - earth, 2
- .C, 17
- attach, 55
- bagEarth, 31
- case.names, 19
- cbind, 12
- contr.earth.response, 1, 14
- contrasts, 2, 14, 24
- data.frames, 12
- density, 47–50
- double, 17
- drop1, 25, 31
- earth, 2, 28, 30, 32–39, 41, 43, 45, 46, 49, 50, 54, 56–63
- etitanic, 15, 26, 27
- evimp, 19, 21, 25, 26, 28, 45, 46, 58, 61
- factor, 9
- factors, 3
- family, 15
- fda, 19
- format.earth, 7, 26, 32, 60, 61
- formula, 55
- gbm, 24
- gc, 17
- glm, 3, 9, 12, 13, 15, 16, 36, 42, 46, 49, 52, 59, 60
- gsub, 55
- hist, 47–50
- image, 53
- importance, 31
- lda, 46, 49
- leaps, 6, 25
- legend, 48
- lm, 5, 14, 36, 42, 46, 49, 59
- lm.fit, 13, 16, 36
- loess, 40, 42
- logical, 9
- mars, 11, 25, 34, 35
- mars.to.earth, 11, 19, 26, 34
- mda, 19
- median, 52
- memory.size, 17
- model.frame.default, 55
- model.matrix.earth, 7, 26, 36
- options, 14
- ozone, 25

ozone1, 26, 37

pairs, 21

par, 43, 55

persp, 53, 54

plot, 39, 43

plot.density, 48–50

plot.earth, 19, 26, 39, 45, 50, 56

plot.earth.models, 26, 43, 43, 45, 46, 56

plot.evimp, 32, 45

plot.histogram, 49, 50

plotd, 26, 42, 43, 45, 46, 56

plotmo, 12, 19, 21, 26, 29, 31, 43, 45, 46, 50, 51

pmax, 24, 33

predict, 24, 47, 49, 52, 54, 55, 58

predict.earth, 9, 24, 26, 47, 52, 53, 56

predict.glm, 47, 57

predict.lm, 53

print.evimp, 58

print.summary.earth  
(summary.earth), 60

randomForest, 31

remove, 17

resid, 59

resid.earth(residuals.earth), 59

residuals, 59

residuals.earth, 26, 59

sample, 20

Scale, 3

scale, 12

set.seed, 18

sqrt, 28

summary, 22

summary.earth, 10, 17–19, 26, 60, 60

termplot, 54, 56, 58

treatment contrast, 14

trees, 29

update, 62

update.earth, 4, 6, 17, 19, 25, 26, 31, 35, 62

varImp, 31